

Chapitre 3 :

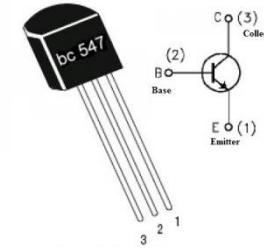
Algèbre de Boole

&

Logiques Combinatoire et Séquentielle

Introduction

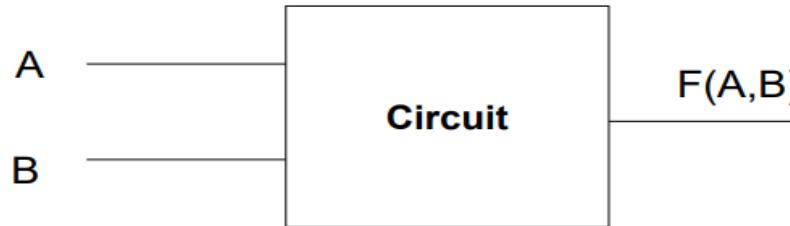
- Tout ordinateur est conçu à partir de **circuits intégrés** qui ont tous une **fonction spécialisée** (ALU, mémoire, circuit décodant les instructions etc.)



- Ces circuits sont fait à partir de **circuits logiques** dont le but est d'exécuter des opérations sur des variables logiques (binaires)

Introduction

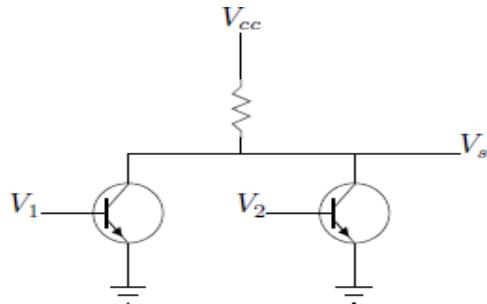
- Chaque circuit fournit une fonction logique bien déterminée; opérations logiques ou arithmétiques (addition, soustraction, comparaison ,....).



- Il est défini par **l'interconnexion** d'un ensemble de **portes logiques** mettant en relation des sorties avec des entrées.
- Une **fonction logique de base** est réalisée à l'aide des **portes logiques** qui permettent d'effectuer des opérations élémentaires.

Introduction

- Ces **portes logiques** sont aujourd'hui réalisées à l'aide de transistors.



- Pour **concevoir et réaliser** ce circuit on doit avoir un **modèle mathématique** de la fonction réalisée par ce circuit .
- Ce modèle doit prendre en considération le **système binaire**.
- Le modèle mathématique utilisé est celui de **Boole**.

Introduction

- 1854 : Georges Boole propose une **algèbre**
- Il a fait des travaux dont lesquels les fonctions (expressions) sont constitués par des variables qui peuvent prendre les valeurs '**OUI**' ou '**NON**' .
- Étude **bien adaptés** au systèmes binaires :
 - Possédant **deux états s'excluant mutuellement**
 - Le système peut être uniquement dans deux états E1 et E2 tel que E1 **est l'opposé de** E2.
 - Le système **ne peut pas être** dans l'état E1 et E2 **en même temps**
- C'est le cas des systèmes numériques (des sous ensembles : les circuits logiques)

Introduction

□ Exemple de systèmes à deux états

- Un interrupteur est ouvert ou non ouvert (fermé)
- Une lampe est allumée ou non allumée (éteinte)
- Une porte est ouverte ou non ouverte (fermée)

□ Remarque :

- En électronique les deux états d'une variable booléenne sont associés à deux niveaux de tension: $V(0)$ et $V(1)$ pour les **états 0 et 1** respectivement.
- On distingue les logiques positive et négative selon que $V(1) > V(0)$ ou $V(1) < V(0)$

Définition et convention

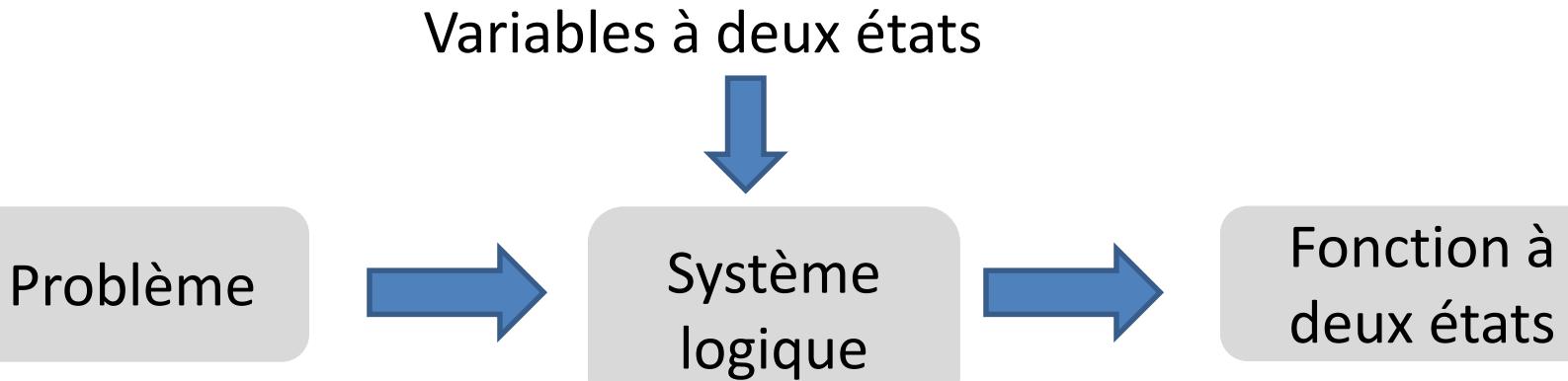
Niveau logique : Lorsque on fait l'étude d'un système logique il faut bien préciser le niveau du travail.

Niveau	Logique positive	Logique négative
Haut	1	0
Bas	0	1

Exemple :

- **Logique positive :**
 - lampe allumée : 1
 - lampe éteinte : 0
- **Logique négative:**
 - lampe allumée : 0
 - lampe éteinte : 1

Concepts de base



Nécessite la maîtrise d'une méthode de conception

1. S'assurer qu'il s'agit d'un système logique
2. Identifier les **variables** booléennes (entrées)
3. Identifier les **fonctions** booléennes (sorties) (**Table de vérité**)
4. Identifier la relation entre les fonctions et les variables (**Equation logique**)
5. Réaliser un schéma logique pour chaque fonction (**Logigramme**)
6. Construire le circuit physique

Variables logiques (booléennes)

- Un système binaire est un système qui ne peut exister que dans **deux états** autorisés.
- Diverses notations peuvent être utilisées pour représenter ces deux états :
 - numérique : 1 et 0
 - logique : vrai et faux
 - électronique : ON et OFF, haut et bas
- **Une variable logique** est une variable qui peut prendre deux états ou valeurs: vrai (V) ou faux (F)
- En faisant correspondre **V** avec le chiffre binaire **1** et **F** avec **0**, ce type de variable devient une variable booléenne ou binaire

Opérateurs logiques de base

Portes logiques

- Une porte logique est un **circuit électronique élémentaire** permettant de réaliser la fonction d'un opérateur logique.

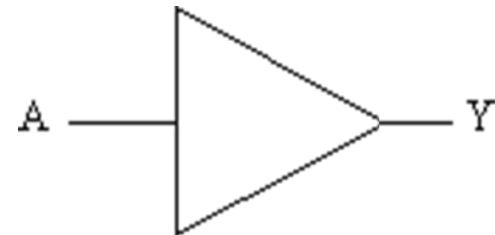
Porte OUI

- OUI: est un opérateur unaire (une seule variable) basique qui à pour rôle **de copier** l'état du bit d'entrée sur le bit de sortie.
- Une seule entrée et une seule sortie
- La sortie d'une fonction OUI prend l'état 1 si et seulement si son entrée est dans l'état 1
- $F(A) = A$

A	$Y = A$
0	0
1	1

table de vérité

Symbolic usual



Symbolic normalisé

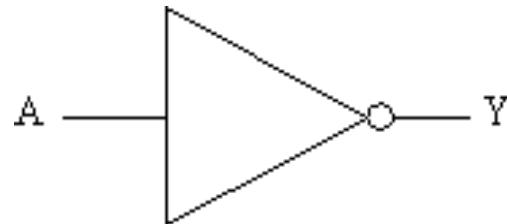


Inverseur : porte NON (négation)

- NON : est un opérateur unaire (une seule variable) qui à pour rôle **d'inverser** la valeur d'une variable .
- Une seule entrée et une seule sortie
- La sortie d'une fonction NON prend l'état 1 si et seulement si son entrée est dans l'état 0
- $F(A) = \text{Non } A = \overline{A}$

A	$Y = \overline{A}$
0	1
1	0

Symbolic usual



Symbolic normalisé



Porte OU (OR)

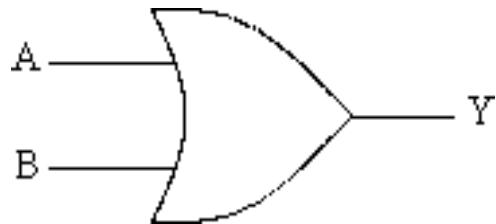
- Le OU est un opérateur binaire (deux variables) , à pour rôle de réaliser la **somme logique** entre **deux variables** logiques.
- Au moins deux entrées
- Le OU fait la **disjonction** entre deux variables.
- La sortie d'une fonction OU est dans l'état 1 si au moins une de ses entrées est dans l'état 1
- Le OU est défini par **$F(A,B)= A + B$** (*il ne faut pas confondre avec la somme arithmétique*)

Porte OU (OR)

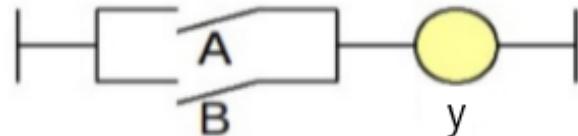
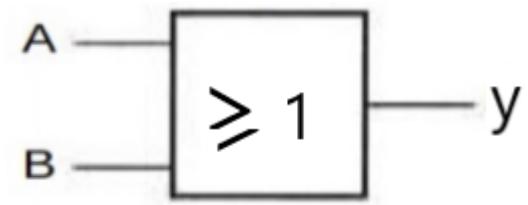
table de vérité

A	B	$Y = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

Symbole usuel



Symbole normalisé



Porte ET (AND)

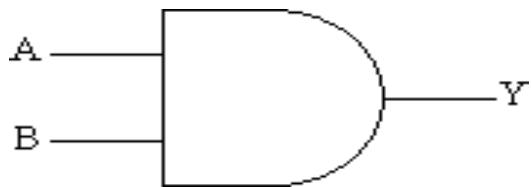
- Le ET est un opérateur binaire (deux variables) , à pour rôle de réaliser le **produit logique** entre **deux variables booléennes**.
- Le ET fait la **conjonction** entre deux variables.
- Au moins deux entrées
- La sortie d'une fonction AND est dans l'état 1 si et seulement si toutes ses entrées sont dans l'état 1
- Le ET est défini par : **$F(A,B)= A \cdot B$**

Porte ET (AND)

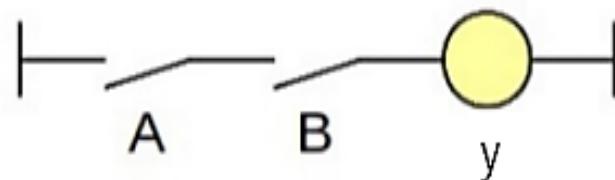
A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

table de vérité

Symbole usuel



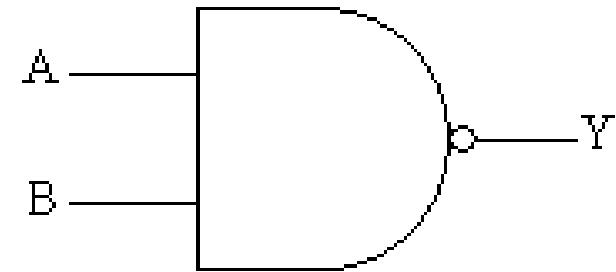
Symbole normalisé



Porte NON ET (NAND)

- Est constituée par un inverseur à la sortie d'une porte ET

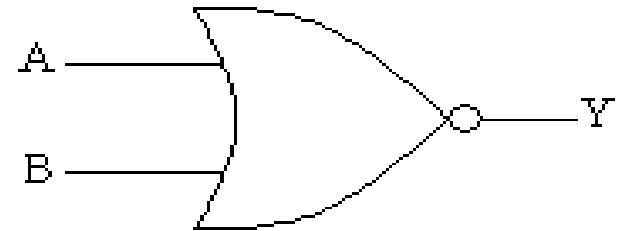
A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0



Porte NON OU (NOR)

- Une négation à la sortie d'une porte OU constitue une fonction NON OU (NOR : NOT OR)

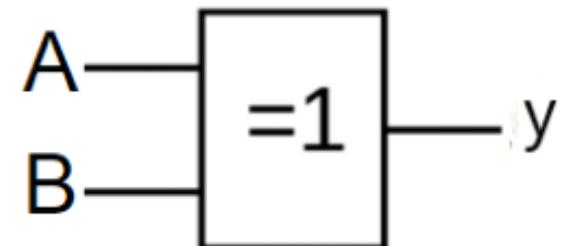
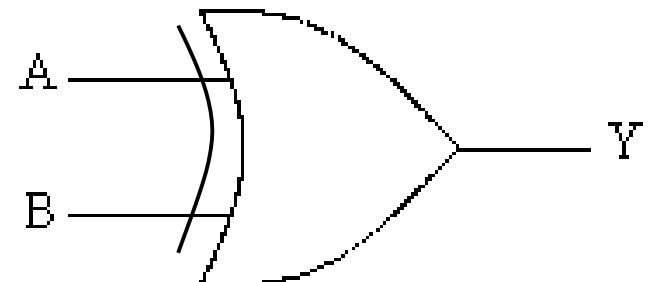
A	B	$Y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0



Porte OU-EXCLUSIF (XOR)

- Au moins deux entrées
- La sortie d'une fonction XOR est dans l'état 1 si le nombre de ses entrées à 1 est un nombre impair

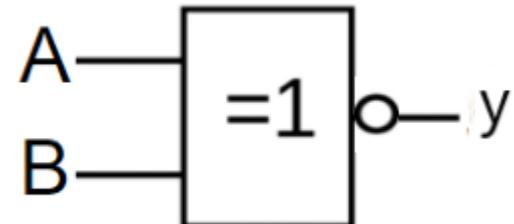
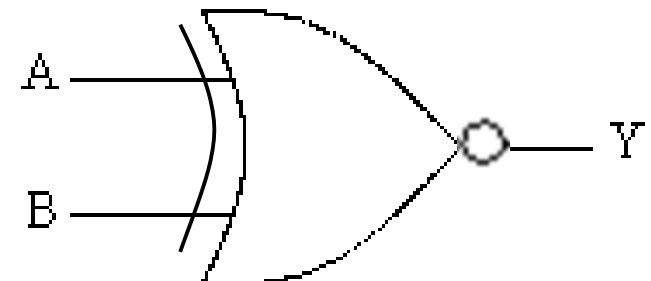
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



Porte NON OU-EXCLUSIF (XNOR)

- Au moins deux entrées
- La sortie d'une fonction XNOR est dans l'état 1 si les deux variables d'entrée sont égales

A	B	$Y = \overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1



Lois fondamentales de l'Algèbre de Boole

Commutativité	Associativité	Distributivité
$a+b = b+a$ $ab = ba$ $a \oplus b = b \oplus a$	$a+(b+c) = (a+b)+c = a+b+c$ $a(bc) = (ab)c = abc$ $a \oplus (b \oplus c) = (a \oplus b) \oplus c = a \oplus b \oplus c$	$a(bc) = (a+b)(a+c)$ $a(b+c) = (ab)+(ac) = ab+ac$ $a(b \oplus c) = (ab) \oplus (ac) = ab \oplus ac$
Élément neutre	Élément absorbant	Idempotence
$a+0 = a$ $1 \cdot a = a$ $a \oplus 0 = a$	$a+1 = 1$ $0 \cdot a = 0$	$a+a = a$ $a \cdot a = a$
Complémentaire	Lois de De Morgan	Divers
$a + \bar{a} = 1$; $a\bar{a} = 0$ $\overline{aa} = \bar{a}$; $\overline{a+a} = \bar{a}$ $\bar{\bar{a}} = a$ $a \oplus \bar{a} = 1$ $a \oplus 1 = \bar{a}$	$\overline{ab} = \bar{a} + \bar{b}$ $\overline{a+b} = \bar{a}\bar{b}$	$a + ab = a(a+b) = a$ $a + (\bar{a}b) = a + b$; $a(\bar{a} + b) = ab$ $a \oplus a = 0$ $a \oplus \bar{b} = \bar{a} \oplus b = \overline{a \oplus b}$ $\bar{a} \oplus \bar{b} = a \oplus b$ $a \oplus b = a\bar{b} + \bar{a}b$ $\overline{a \oplus b} = ab + \bar{a}\bar{b}$

Lois fondamentales de l'Algèbre de Boole

- Démonstration du théorème de l'inhibition:

$$x + (\bar{x}y) = (x + \bar{x}).(x + y)$$

Distributivité

$$= 1.(x + y)$$

Complémentarité

$$= x + y$$

Elément neutre

$$\boxed{x + (\bar{x}y) = x + y}$$

Lois fondamentales de l'Algèbre de Boole

- Démonstration du théorème de l'absorption :

$$\begin{aligned} X+1 &= x + (x + \bar{x}) \\ &= (x + x) + \bar{x} \\ &= x + \bar{x} \\ &= 1 \end{aligned}$$

Complémentarité

associativité

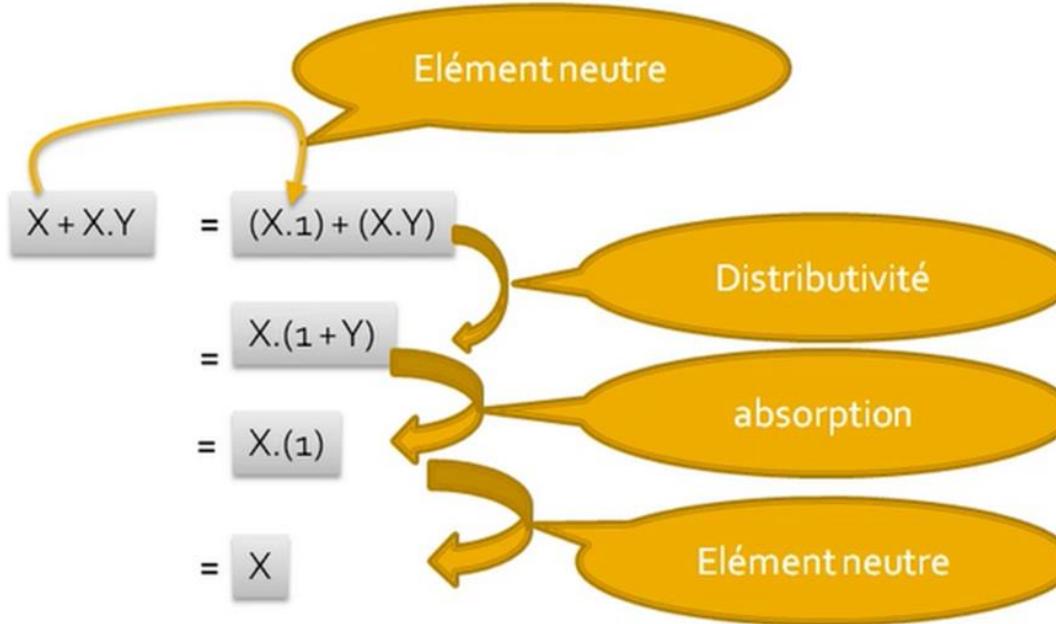
Idempotence

The diagram illustrates the proof of the absorption law in Boolean algebra. It shows the simplification of the expression $X+1$ through three steps:

- Step 1:** $X+1 = x + (x + \bar{x})$ (Complémentarité: A variable plus its complement equals 1)
- Step 2:** $= (x + x) + \bar{x}$ (associativité: The sum operation is associative)
- Step 3:** $= x + \bar{x}$ (Idempotence: A variable plus itself equals the variable)
- Final Step:** $= 1$

Lois fondamentales de l'Algèbre de Boole

■ Démonstration du théorème de l'absorption :



D'où $X + X.Y = X$

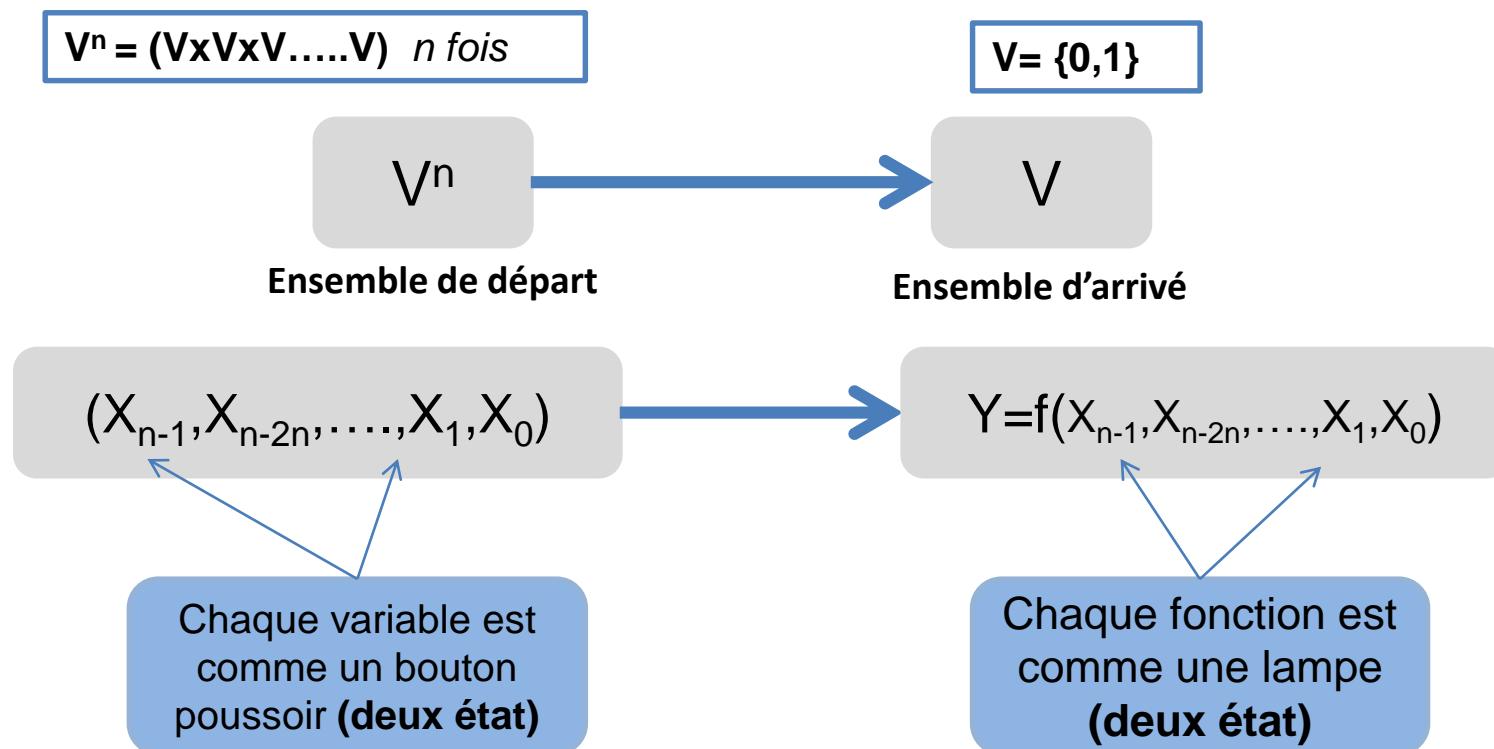
Fonctions logiques (booléennes)

- Une fonction booléenne à une ou plusieurs variables est une fonction qui **renvoie une valeur ne dépendant que de ces variables**
- Une fonction booléenne est une fonction définie sur 2^n combinaisons de n variables logiques.
- Une fonction logique peut prendre en sortie 2 valeurs notées 0 et 1.
- Elle peut être complètement décrite par une table à 2^n lignes donnant la valeur de la fonction pour chaque combinaison d'entrées
 - Table de vérité de la fonction

Fonctions logiques (booléennes)

- Toute fonction logique peut être réalisée à l'aide des **portes**
- Réalisation d'une fonction booléenne
 - Écrire l'équation de la fonction à partir de sa table de vérité
 - Simplifier l'équation
 - Réaliser l'équation à l'aide des portes disponibles

Fonctions logiques (booléennes)



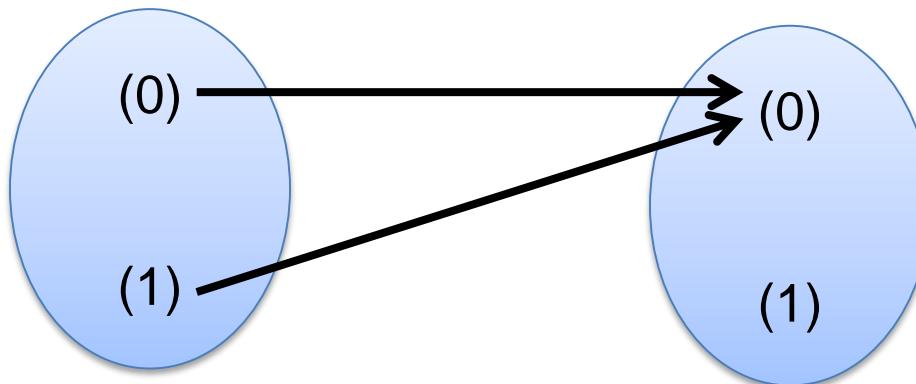
Concrètement, elle correspond à une action comme:

- Allumer une lampe
- Démarrer un moteur
- Eteindre un moteur
- Etc...

Fonctions booléennes

□ Fonctions logique à une variable

→ Combien de fonction on peut avoir avec une seule variable ?



$$F_0(x) = 0$$

Fonction constante

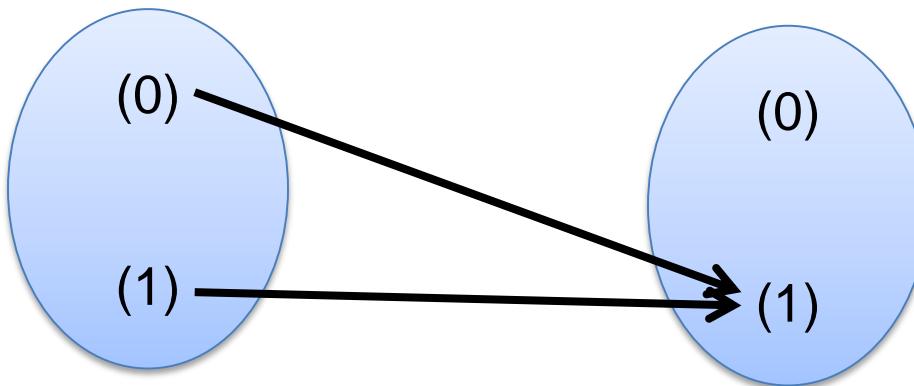
Ensemble de
départ : V

Ensemble
d'arrivée: V

Fonctions booléennes

□ Fonctions logique à une variable

→ Combien de fonction on peut avoir avec une seule variable ?



$$F_1(x) = 1$$

Fonction constante

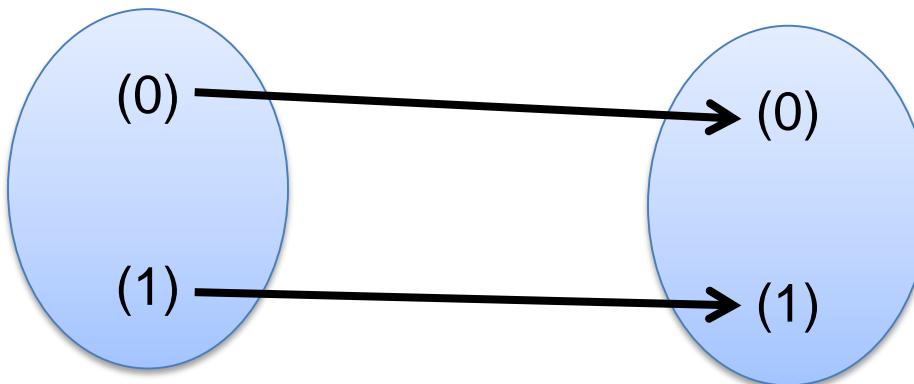
Ensemble de
départ : V

Ensemble
d'arrivée: V

Fonctions booléennes

□ Fonctions logique à une variable

→ Combien de fonction on peut avoir avec une seule variable ?



$$F_2(x) = x$$

Fonction identité

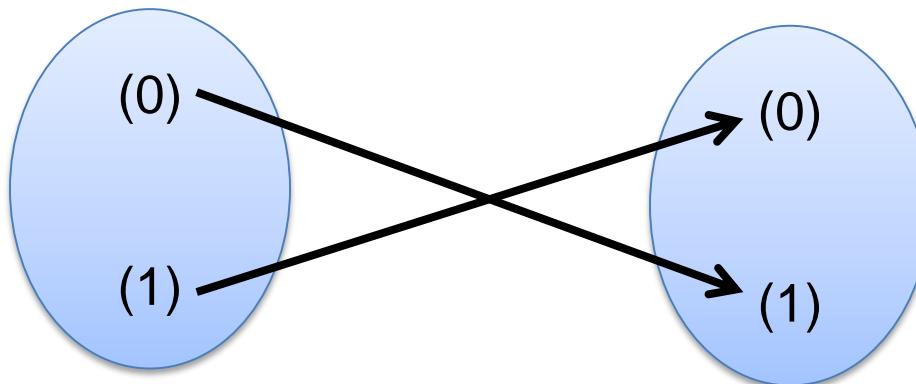
Ensemble de départ : V

Ensemble d'arrivée: V

Fonctions booléennes

□ Fonctions logique à une variable

→ Combien de fonction on peut avoir avec une seule variable ?



$$F_3(x) = \bar{x}$$

Fonction négation

→ Uniquement 4 fonctions possibles !

Fonctions booléennes

□ Fonctions logique à une variable

- Pour représenter les fonctions → Table de vérité

		Les variables		Les fonctions		
		x		$F_0(x) = 0$	$F_1(x) = 1$	
$n=1$			0	0	1	
	2^n situations		1	0	1	
				$F_2(x) = x$	$F_3(x) = \bar{x}$	
		0	1	0	1	
		1	0	1	0	
Fonction constante			Fonction identité		Fonction négation	

Fonctions booléennes

□ Fonctions logique à 2 variables

$$V^n = (V \times V) \text{ 2 fois}$$

$$V = \{0,1\}$$

$$V^2$$

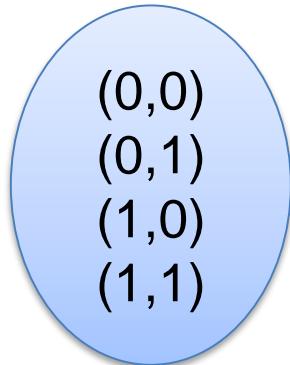
Ensemble de départ

$$V$$

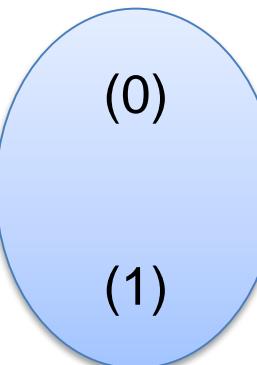
Ensemble d'arrivée

$$(X_1, X_0)$$

$$Y = f(X_1, X_0)$$



Ensemble de départ : V^2



Ensemble d'arrivée: V

Fonctions booléennes

□ Fonctions logique à 2 variables

Les fonctions

2 variables		Les fonctions															
X_1	X_0	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

2² = 4 situations

Fonction constante **Fonction ET** **Fonction OU** **Fonction constante**

Fonctions booléennes

□ Mettre en équation une fonction ?

- Soit x une variable :
- Si $x = 0$ on écrit \bar{x}
 - Si $x = 1$ on écrit x

x_1	x_0	$F(x_1, x_0)$
-------	-------	---------------

minterme	m_0	0	0	V_0	Situation 0	$(F=V_0) \text{ ET } (x_1=0) \text{ ET } (x_0=0)$	$V_0 \cdot \bar{x}_1 \cdot \bar{x}_0$	$=V_0 m_0$
	m_1	0	1	V_1	Situation 1	$(F=V_1) \text{ ET } (x_1=0) \text{ ET } (x_0=1)$	$V_1 \cdot \bar{x}_1 \cdot x_0$	$=V_1 m_1$
	m_2	1	0	V_2	Situation 2	$(F=V_2) \text{ ET } (x_1=1) \text{ ET } (x_0=0)$	$V_2 \cdot x_1 \cdot \bar{x}_0$	$=V_2 m_2$
	m_3	1	1	V_3	Situation 3	$(F=V_3) \text{ ET } (x_1=1) \text{ ET } (x_0=1)$	$V_3 \cdot x_1 \cdot x_0$	$=V_3 m_3$

4 situations(disjointes)
pour 2 variables

Fonctions booléennes

□ Mettre en équation une fonction ?

A un moment donné: $F(x_1, x_0) = v_i m_i$
i allant de 0 à 3

Forme algébrique
canonique disjonctive



Donc $F(x_1, x_0) = v_0 m_0 \text{ OU } v_1 m_1 \text{ OU } v_2 m_2 \text{ OU } v_3 m_3$



Ce qui donne :

$$F(x_1, x_0) = v_0 m_0 + v_1 m_1 + v_2 m_2 + v_3 m_3$$

$$F(x_1, x_0) = \sum_{i=0}^3 (v_i \cdot m_i)$$

Fonctions booléennes

□ Mettre en équation une fonction ?

Trouvons l'équation de $f(x,y)$ définie par la TV suivante

$$F(x,y) = v_0m_0 + v_1m_1 + v_2m_2 + v_3m_3$$

x	y	$f(x,y)$
0	0	$v_0 = 0$
0	1	$v_1 = 1$
1	0	$v_2 = 1$
1	1	$v_3 = 0$

$$m_0 = (\bar{x} \cdot \bar{y})$$

$$m_1 = (\bar{x} \cdot y)$$

$$m_2 = (x \cdot \bar{y})$$

$$m_3 = (x \cdot y)$$

$$F(x, y) = (\bar{x} \cdot \bar{y}) \cdot v_0 + (\bar{x} \cdot y) \cdot v_1 + (x \cdot \bar{y}) \cdot v_2 + (x \cdot y) \cdot v_3$$

$$F(x, y) = (\bar{x} \cdot \bar{y})^0 + (\bar{x} \cdot y)^1 + (x \cdot \bar{y})^1 + (x \cdot y)^0$$

$$F(x, y) = (\bar{x} \cdot y) + (x \cdot \bar{y})$$

Elément neutre

Elément absorbant

Fonctions booléennes

□ Mettre en équation une fonction ?

Trouvons l'équation de $f(x,y,z)$ définie par la TV suivante

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Valeurs de f

v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7

Mintermes

m_0
m_1
m_2
m_3
m_4
m_5
m_6
m_7

Termes associés

$\bar{x} \cdot \bar{y} \cdot \bar{z}$
$\bar{x} \cdot \bar{y} \cdot z$
$\bar{x} \cdot y \cdot \bar{z}$
$\bar{x} \cdot y \cdot z$
$x \cdot \bar{y} \cdot \bar{z}$
$x \cdot \bar{y} \cdot z$
$x \cdot y \cdot \bar{z}$
$x \cdot y \cdot z$

$$\rightarrow F(x,y,z) = m_0 v_0 + m_1 v_1 + m_2 v_2 + m_3 v_3 + m_4 v_4 + m_5 v_5 + m_6 v_6 + m_7 v_7$$

$$\rightarrow F(x,y,z) = m_0 \cdot 0 + m_1 \cdot 1 + m_2 \cdot 1 + m_3 \cdot 0 + m_4 \cdot 0 + m_5 \cdot 0 + m_6 \cdot 0 + m_7 \cdot 1$$

$$\rightarrow F(x,y,z) = m_1 + m_2 + m_7$$

$$\rightarrow F(x, y, z) = \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot \bar{z} + x \cdot y \cdot z$$

$$F(x,y,z) = m_1 + m_2 + m_7 = \sum(1, 2, 7)$$

Réalisation des fonctions booléennes

- À partir de la table de vérité, nous pouvons avoir deux formes analytiques, dénommées formes canoniques
 - somme canonique de produits (**SOP**)
 - Forme canonique disjonctive
 - produit canonique de sommes (**POS**)
 - Forme canonique conjonctive

Écritures canoniques (SOP)

- **3 variables**, terme produit, qu'on appelle **minterme**, égal au **ET** des variables qui composent cette combinaison

	x	y	z	$\overline{x}\overline{y}z$	$\overline{x}\overline{y}z$	$\overline{x}\overline{y}\overline{z}$	$\overline{x}yz$	$\overline{x}\overline{y}z$	$\overline{x}\overline{y}\overline{z}$	$\overline{x}yz$	$\overline{x}\overline{y}\overline{z}$	$x\overline{y}z$
0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0	0
6	1	1	0	0	0	0	0	0	0	1	0	0
7	1	1	1	0	0	0	0	0	0	0	0	1

Écritures canoniques (SOP)

A	B	C	F	$P_3 + P_5 + P_6 + P_7$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Cette façon, très générale,
d'écrire une fonction
booléenne est appelée somme
canonique de produits (SOP)

$$F(A, B, C) = P_3 + P_5 + P_6 + P_7$$



$$F(A, B, C) = \overline{A}BC + A\overline{B}C + ABC + A\overline{B}\overline{C} = \sum(3, 5, 6, 7)$$

→ La forme disjonctive.

Écritures canoniques (POS)

- **3 variables**, terme somme, qu'on appelle **maxterme**, égal au **OU** des variables qui composent cette combinaison

				S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
	X	Y	Z	$X+Y+Z$	$X+Y+\bar{Z}$	$X+\bar{Y}+Z$	$X+\bar{Y}+\bar{Z}$	$\bar{X}+Y+Z$	$\bar{X}+Y+\bar{Z}$	$\bar{X}+\bar{Y}+Z$	$\bar{X}+\bar{Y}+\bar{Z}$
0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
2	0	1	0	1	1	0	1	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1
4	1	0	0	1	1	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0	1	1
6	1	1	0	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	0

Écritures canoniques (POS)

$$F(X, Y, Z) = S_0 \cdot S_1 \cdot S_2 \cdot S_4$$



$$F(X, Y, Z) = (X + Y + Z) \cdot (X + Y + \overline{Z}) \cdot (X + \overline{Y} + Z) \cdot (\overline{X} + Y + Z)$$

$$F(X, Y, Z) = \prod S(0,1,2,4)$$

X	Y	Z	F	$S_0 \cdot S_1 \cdot S_2 \cdot S_4$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Cette écriture est appelée
produit canonique de sommes
(POS)

→ La forme conjonctive.

Écritures canoniques POS/SOP

A	B	C	S	
0	0	0	0	→ A + B + C : max terme
0	0	1	0	→ A + B + \bar{C} : max terme
0	1	0	0	→ A + \bar{B} + C : max terme
0	1	1	1	→ $\bar{A} \cdot B \cdot C$: min terme
1	0	0	0	→ $\bar{A} + B + C$: max terme
1	0	1	1	→ A $\cdot \bar{B} \cdot C$: min terme
1	1	0	1	→ A $\cdot B \cdot \bar{C}$: min terme
1	1	1	1	→ A $\cdot B \cdot C$: min terme

Écritures canoniques

- Écritures canoniques expriment une fonction booléenne à l'aide des opérateurs logiques ET, OU, NON



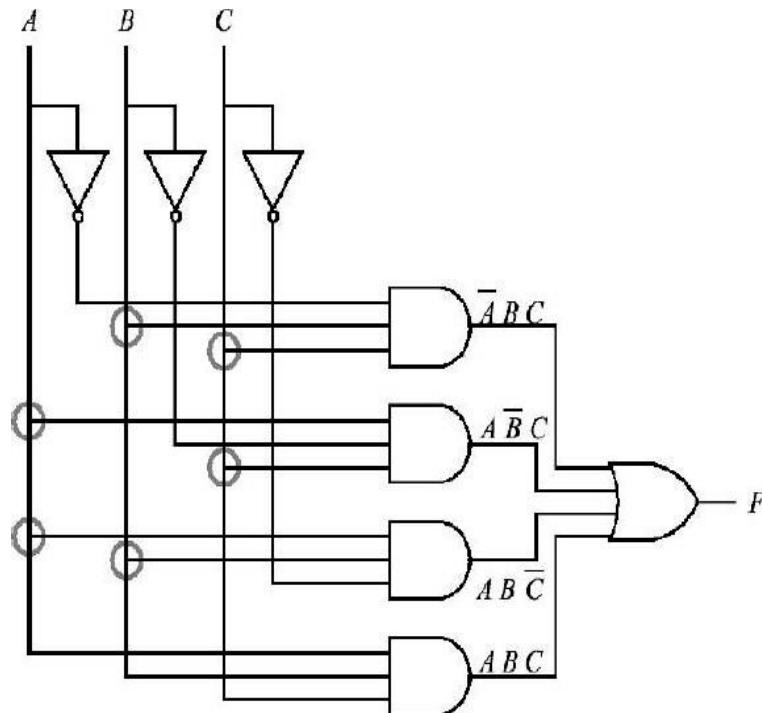
**On peut réaliser une fonction
à l'aide des portes ET, OU, NON**

Écritures canoniques d'une fonction logique

\overline{ABC}

→

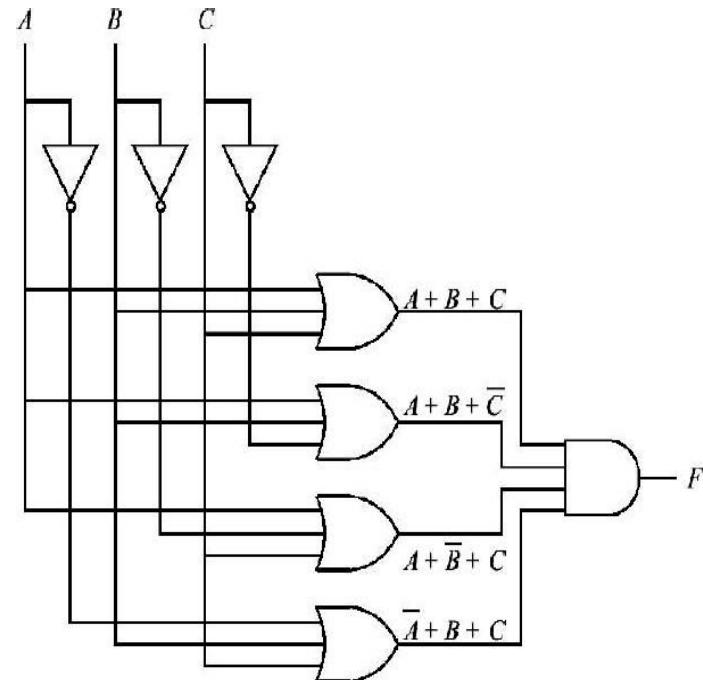
A	B	C	F	$P_3 + P_5 + P_6 + P_7$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



Écritures canoniques d'une fonction logique

$A+B+C$

A	B	C	F	$S_0 \cdot S_1 \cdot S_2 \cdot S_4$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



Exercice

Trouver la table de vérité de la fonction:

$$F(A, B, C) = (\overline{A \cdot B}) \cdot (C + B) + A \cdot \overline{B} \cdot C$$

Exemple:

si on veut calculer $F(0,1,1)$ alors :

$$F(0,1,1) = (\overline{0 \cdot 1})(1 + 1) + 0 \cdot \bar{1} \cdot 1$$

$$F(0,1,1) = (\overline{0})(1) + 0 \cdot 0 \cdot 1$$

$$F(0,1,1) = 1 \cdot 1 + 0 \cdot 0 \cdot 1$$

$$F(0,1,1) = 1 + 0$$

$$F(0,1,1) = 1$$

Exercice

Trouver la table de vérité de la fonction: $F(A, B, C) = (\overline{A \cdot B}) \cdot (\overline{C} + B) + A \cdot \overline{B} \cdot C$

Solution

- Pour trouver la table de vérité , il faut trouver la valeur de la fonction F pour chaque combinaisons des trois variables A, B , C
- 3 variables $\rightarrow 2^3 = 8$ combinaisons

$$F(A, B, C) = (\overline{A \cdot B}) \cdot (\overline{C} + B) + A \cdot \overline{B} \cdot C$$

$$F(0,0,0) = (\overline{0 \cdot 0}) \cdot (0 + 0) + 0 \cdot \overline{0} \cdot 0 = 0$$

$$F(0,0,1) = (\overline{0 \cdot 0}) \cdot (1 + 0) + 0 \cdot \overline{0} \cdot 1 = 1$$

$$F(0,1,0) = (\overline{0 \cdot 1}) \cdot (0 + 1) + 0 \cdot \overline{1} \cdot 0 = 1$$

$$F(0,1,1) = (\overline{0 \cdot 1}) \cdot (1 + 1) + 0 \cdot \overline{1} \cdot 1 = 1$$

$$F(1,0,0) = (\overline{1 \cdot 0}) \cdot (0 + 0) + 1 \cdot \overline{0} \cdot 0 = 0$$

$$F(1,0,1) = (\overline{1 \cdot 0}) \cdot (1 + 0) + 1 \cdot \overline{0} \cdot 1 = 1$$

$$F(1,1,0) = (\overline{1 \cdot 1}) \cdot (0 + 1) + 1 \cdot \overline{1} \cdot 0 = 0$$

$$F(1,1,1) = (\overline{1 \cdot 1}) \cdot (1 + 1) + 1 \cdot \overline{1} \cdot 1 = 0$$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Relation d'équivalence des circuits

- Soucis majeurs des concepteurs
 - Réduire le nombre de portes nécessaires à la réalisation des systèmes
 - Minimiser le coût en nombre de boîtiers
 - La consommation électrique
 - Réduction du temps de parcours du signal
 - Minimiser la complexité
 - Créer un système équivalent avec certains paramètres optimisés
 - Recherche d'équivalence
 - Utiliser les lois et théorèmes de l'algèbre de Boole

Relation d'équivalence des circuits

- La plupart des règles de l'algèbre ordinaire restent valides pour l'algèbre booléenne
- **Exemple** : $A \cdot B + A \cdot C = A \cdot (B + C)$
 - On passe de trois portes à deux
 - Le nombre de niveaux de portes reste le même

Relation d'équivalence des circuits

□ Trouver l'équation de la fonction Y ?

	A	B	$Y = A \oplus B$
m_0	0	0	0
m_1	0	1	1
m_2	1	0	1
m_3	1	1	0

$$Y=F(A,B) = m_0 \cdot 0 + m_1 \cdot 1 + m_2 \cdot 1 + m_3 \cdot 0$$
$$A \oplus B = m_1 + m_2$$

$$A \oplus B = \overline{A} \bullet B + A \bullet \overline{B}$$

Relation d'équivalence des circuits

□ Exemple de manipulation algébrique:

$$F(A, B, C) = \overline{\overline{A} \cdot \overline{B} \cdot C} + \overline{\overline{A} \cdot B \cdot \overline{C}} + A \cdot \overline{\overline{B} \cdot \overline{C}} + A \cdot B \cdot C =$$

$$= C \cdot (\overline{\overline{A} \cdot \overline{B}} + A \cdot B) + \overline{C} \cdot (A \cdot \overline{\overline{B}} + \overline{A} \cdot B) =$$

$$= C \cdot (\overline{A \oplus B}) + \overline{C} \cdot (A \oplus B) = A \oplus B \oplus C$$

Relation d'équivalence des circuits

- Deux fonctions logiques **sont équivalentes** si, et seulement si, les **valeurs de leurs sorties** sont les **mêmes** pour toutes les configurations identiques de leurs variables d'entrée
→ Examen des tables de vérité respectives

Système Logique Complet (SLC)

- Toute fonction booléenne d'un nombre quelconque de variables peut s'écrire avec les trois fonctions de base **ET**, **OU** et **NON**

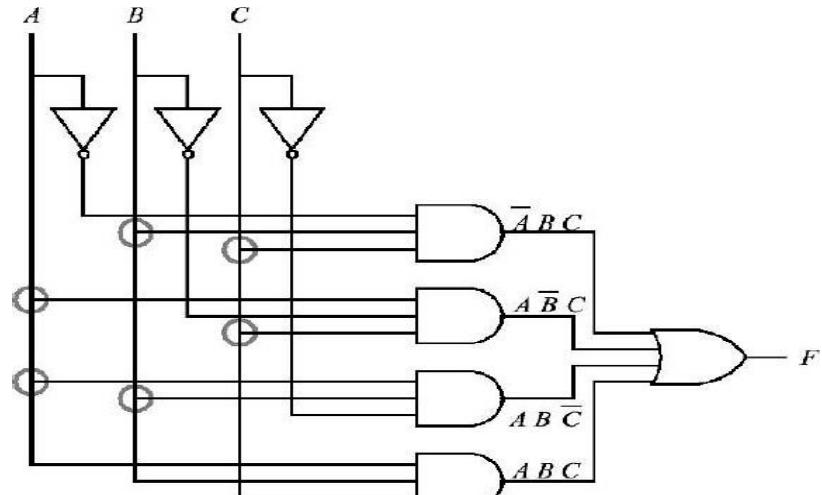
→ L'ensemble { ET, OU, NON } est un **SLC Système Logique Complet**

- Il existe d'autres SLC :

{ET,NON} | **{OU,NON}** | **NON-ET (NAND)** | **NON-OU (NOR)**

- Exemple:**

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



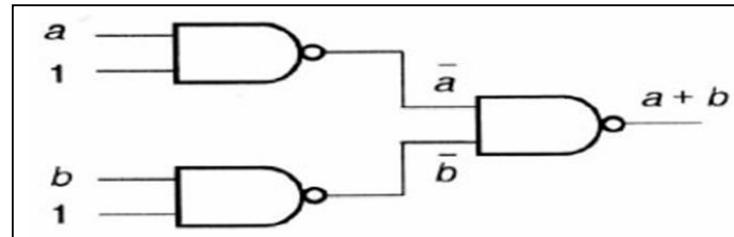
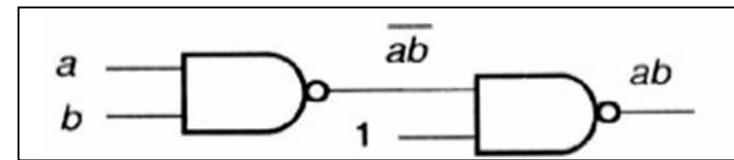
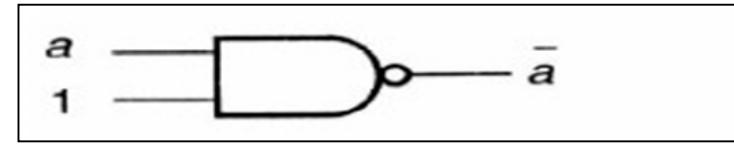
Ensemble { NON-ET (NAND) } et {ET,NON}

- { NON-ET (NAND) } est complet et minimal
- Les portes NON, OU et ET peuvent être obtenues à partir de portes NON-ET

{ET,NON}

$$A+B = \overline{\overline{A}+\overline{B}} = \overline{\overline{A}} \cdot \overline{\overline{B}}$$

NON-ET (NAND)



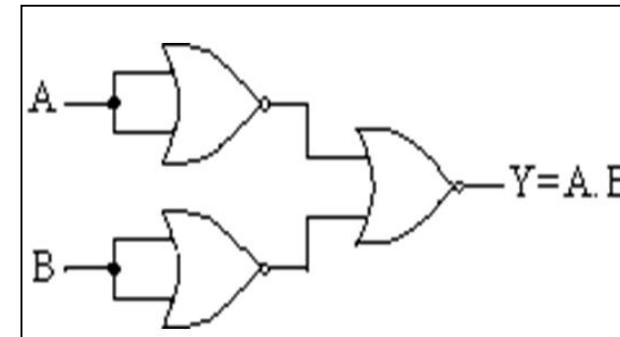
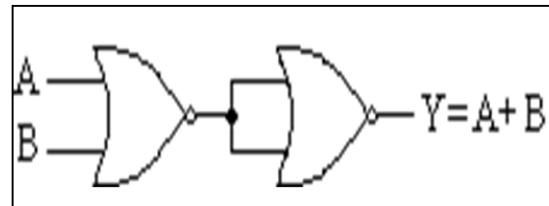
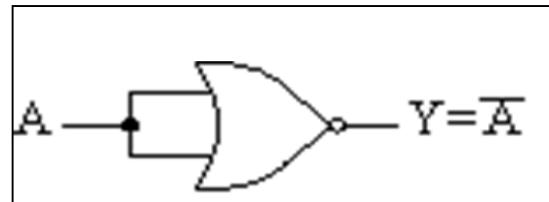
Ensemble { NON-OU (NOR) } et {OU,NON}

- { NON-OU (NOR) } est complet et minimal
- Les portes NON, OU et ET peuvent être obtenues à partir de portes NON-OU

{OU,NON}

$$A \cdot B = \overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A} + \overline{B}}$$

NON-OU (NOR)



Synthèse d'un circuit logique

- Trouver sa fonction logique
- Principe
 - Donner l'expression des sorties de chaque porte/composant en fonction des valeurs de ses entrées
 - En déduire au final la (ou les) fonction(s) logique(s) du circuit
 - On peut ensuite
 - Déterminer la table de vérité du circuit
 - Simplifier la fonction logique

Synthèse d'un circuit logique

- A partir d'une fonction logique trouver le logigramme correspondant à cette fonction
- Principe
 - Simplifier la fonction logique avec les méthodes:
 - La méthode algébrique (algèbre de Boole)
 - La méthode des tableaux de Karnaugh
 - Les méthodes programmables (MC-Cluskey)
 - En déduire le logigramme correspondant

Simplification des fonctions logiques

Simplification /optimisation ?

Méthodes «classiques» de simplifications :

- pas de solution unique
- indépendant de la technologie
- le temps n'est pas pris en compte

La simplification «mathématique» n'est pas toujours optimale en regard des critères d'optimisation technologiques.

Simplification /optimisation ?

- L'objectif de la simplification des fonctions logiques est de :
 - réduire le **nombre de termes** dans une fonction
 - réduire le **nombre de variables** dans un terme
- Cela afin de réduire le nombre de **portes logiques** utilisées
→ réduire le **coût du circuit**
- Plusieurs méthodes existent pour la simplification :
 - 1) **Les méthodes algébriques**
 - 2) **Les méthodes graphiques** : (ex : **tableaux de karnaugh**)
 - 3) **Les méthodes programmables (MC-Cluskey)**

Les méthodes algébriques

- Le principe consiste à appliquer les **règles** de l'algèbre de Boole afin d'éliminer des variables ou des termes.
 - Mais il n'y a pas une **démarche bien spécifique**.
- **Voici quelques règles les plus utilisées :**
 - **Minterme adjacent = 1 seule variable qui change**
 - ❖ Deux minterme adjacents → il reste l'intersection commune
 - ❖ Deux maxterme adjacents → il reste la réunion(somme) commune

Exemple:

$$a.b.c + a.b.\bar{c} = a.b.(c + \bar{c}) = a.b$$
$$(a + b + c).(a + b + \bar{c}) = (a + b)(c + \bar{c}) = a + b$$

- ❖ Rajouter un terme déjà existant à une expression
 - Pas de coefficient en algèbre de Boole
- ❖ On ne change pas le résultat en multipliant l'un des termes par **1** ou en ajoutant **0**.
- ❖ il est possible de supprimer un terme superflu (un terme en plus), c'est-à-dire déjà inclus dans la réunion des autres termes.

Les méthodes algébriques

- **Quelques règles:** $A \cdot B + \overline{A} \cdot B = B$
 $A + A \cdot B = A$
 $A + \overline{A} \cdot B = A + B$
 $(A + B)(A + \overline{B}) = A$
 $A \cdot (A + B) = A$
 $A \cdot (\overline{A} + B) = A \cdot B$

□ Exemple de simplifications

$$S_1 = (A + \overline{B})(\overline{A} + B)(\overline{A} + \overline{B})$$

$$\begin{aligned} S_1 &= (A + \overline{B})(\overline{A} + B)(\overline{A} + \overline{B}) = (A\overline{A} + \overline{B}\overline{A} + A.B + \overline{B}.B).(\overline{A} + \overline{B}) = (0 + \overline{B}\overline{A} + A.B + 0).(\overline{A} + \overline{B}) \\ &= (\overline{B}\overline{A} + A.B).(\overline{A} + \overline{B}) = \overline{B}\overline{A}\overline{A} + A.B\overline{A} + \overline{B}A\overline{B} + A.B\overline{B} = \overline{B}0 + A\overline{A}B + \overline{A}B\overline{B} + A.0 \\ &= A\overline{A}B + \overline{A}\overline{B} = 0.B + \overline{A}\overline{B} = \overline{A}\overline{B} \end{aligned}$$

Les méthodes algébriques

Exercice 1:

Démontrez la proposition suivante :

$$1/ \quad ABC + ABC + A\bar{B}CD = AB + ACD$$

$$2/ \quad A\bar{B}C + \bar{A}BC + A\bar{B}C + ABC = BC + AC + AB$$

$$3/ \quad F(A, B, C) = A\bar{B} + \bar{B}C + AC = A\bar{B} + \bar{B}C$$

Les méthodes algébriques

□ Solution

- 1/

$$\begin{aligned}ABC + AB \bar{C} + A \bar{B}CD &= AB(C + \bar{C}) + A \bar{B}CD \\&= AB + A \bar{B}CD \\&= A(B + \bar{B}(CD)) \\&= A(B + CD) \\&= AB + ACD\end{aligned}$$

- 2/

$$\begin{aligned}A B C + \bar{A} B C + A \bar{B} C + A B \bar{C} &= \\A B C + \bar{A} B C + A B C + A \bar{B} C + A B C + A B \bar{C} &= \\BC + AC + AB\end{aligned}$$

- 3/

$$\begin{aligned}F(A, B, C) &= A B + \bar{B} C + A C = AB + \bar{B} C + AC(B + \bar{B}) \\&= AB + \bar{B} C + ACB + A \bar{B} C \\&= AB(1 + C) + \bar{B} C(1 + A) \\&= AB + \bar{B} C\end{aligned}$$

Les méthodes algébriques

Exercice 2:

Démontrer la proposition suivante :

$$A \cdot B + B \cdot C + A \cdot C + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C = A + B + C$$

Donner la forme simplifiée de la fonction suivante :

$$F(A, B, C, D) = \overline{A} \overline{B} \overline{C} \overline{D} + A \overline{B} \overline{C} \overline{D} + A B \overline{C} \overline{D} + A B C \overline{D} + A B C D$$

La méthode des tableaux de Karnaugh

- Méthode graphiques de simplification
- Le diagramme de Karnaugh d'une fonction logique est une transformation graphique de la table de vérité qui permet la visualisation de tous les mintermes

La méthode des tableaux de Karnaugh

- La méthode consiste à mettre en évidence par une méthode **graphique** (un tableau) tous les termes qui sont **adjacents** (qui ne diffèrent que par l'état d'une seule variable).
- Un tableau de Karnaugh = table de vérité de 2^n cases avec un changement unique entre 2 cases voisines d'où des codes cycliques (Gray ou binaire réfléchi).
- La méthode peut s'appliquer aux fonctions logiques de **2,3,4,5 et 6 variables**.
- Les tableaux de Karnaugh comportent **2^n cases** (n: est le nombre de variables).

La méthode des tableaux de Karnaugh

A 2-variable Karnaugh map with variables A and B. The vertical axis (B) has values 0 and 1. The horizontal axis (A) has values 0 and 1. The four cells are labeled 0, 1, 0, 1 from top-left to bottom-right.

	A	
B	0	1
0		
1		

Tableau à 2 variables

A 3-variable Karnaugh map with variables C, AB. The vertical axis (C) has values 0 and 1. The horizontal axis (AB) has values 00, 01, 11, 10. The eight cells are empty.

	AB			
C	00	01	11	10
0				
1				

Tableaux à 3 variables

A 4-variable Karnaugh map with variables CD, AB. The vertical axis (CD) has values 00, 01, 11, 10. The horizontal axis (AB) has values 00, 01, 11, 10. All 16 cells are empty.

	AB			
CD	00	01	11	10
00				
01				
11				
10				

Tableau à 4 variables

A 5-variable Karnaugh map for U=0 with variables CD, AB. The vertical axis (CD) has values 00, 01, 11, 10. The horizontal axis (AB) has values 00, 01, 11, 10. The 16 cells are empty.

	AB			
CD	00	01	11	10
00				
01				
11				
10				

Tableau à 5 variables

A 5-variable Karnaugh map for U=1 with variables CD, AB. The vertical axis (CD) has values 00, 01, 11, 10. The horizontal axis (AB) has values 00, 01, 11, 10. The 16 cells are empty.

	AB			
CD	00	01	11	10
00				
01				
11				
10				

A 6-variable Karnaugh map with variables ed, cba. The vertical axis (ed) has values 00, 01, 11, 10. The horizontal axis (cba) has values 000, 001, 011, 010, 110, 111, 101, 100. The 32 cells are empty.

	cba							
ed	000	001	011	010	110	111	101	100
00								
01								
11								
10								

La méthode des tableaux de Karnaugh

□ Les termes adjacents

- Examinons l'expression suivante : $A \cdot B + A \cdot \bar{B}$
 - Les deux termes possèdent les mêmes variables. La seule différence est **l'état de la variable B qui change.**
 - Si on applique les règles de simplification on obtient :

$$AB + A\bar{B} = A(B + \bar{B}) = A$$

- Donc ces termes sont **adjacents**
- **Exemple:** Ces termes sont adjacents

$$A \cdot B + \bar{A} \cdot B = B$$

$$A \cdot B \cdot C + A \cdot \bar{B} \cdot C = A \cdot C$$

$$A \cdot B \cdot C \cdot D + A \cdot B \cdot \bar{C} \cdot D = A \cdot B \cdot D$$

Ces termes ne sont pas adjacents

$$A \cdot B + \bar{A} \cdot \bar{B}$$

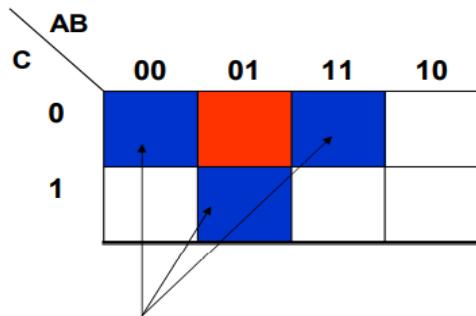
$$A \cdot B \cdot C + A \cdot \bar{B} \cdot \bar{C}$$

$$A \cdot B \cdot C \cdot D + \bar{A} \cdot B \cdot \bar{C} \cdot D$$

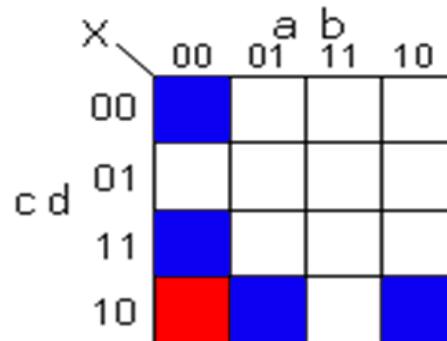
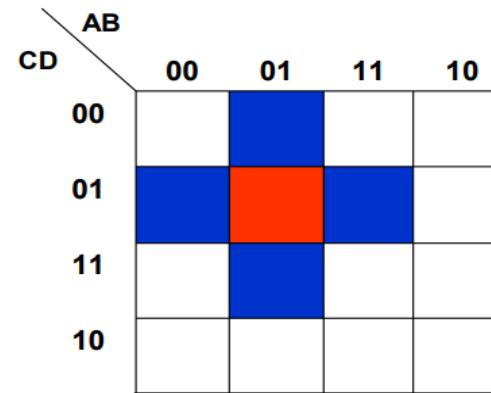
La méthode des tableaux de Karnaugh

□ Les cases adjacentes

Dans un tableau de Karnaugh , chaque case possède un certain nombre de cases adjacentes.



Les trois cases bleues sont des cases adjacentes à la case rouge



La méthode des tableaux de Karnaugh

□ Passage de la table de vérité à la table de Karnaugh

- Pour chaque combinaison qui représente un **min terme**
→ correspond à une **case** dans le tableau qui **doit être mise à 1.**
- Pour chaque combinaison qui représente un **max terme**
→ correspond à une **case** dans le tableau qui **doit être mise à 0.**
- Lorsque on remplit le tableau, on doit prendre soit les min termes ou les max termes

La méthode des tableaux de Karnaugh

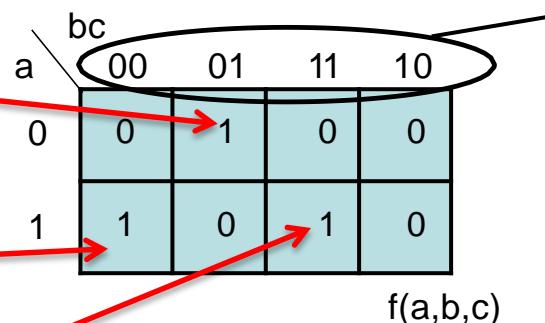
□ Passage de la table de vérité à la table de Karnaugh

$f(a,c,d, \dots, n)$ fonction logique à N entrées sera représentée par:

- une table à 2^N lignes
- un tableau à 2^N cases

Exemple:

a b c	$f(a,b,c)$
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1



Code Gray ou
binaire réfléchi
=
1 seul changement
entre 2 codes
successifs

La méthode des tableaux de Karnaugh

□ Passage de la forme canonique à la table de Karnaugh

Exemple:

$$F1(A,B,C) = \sum (1,2,5,7)$$

		AB	
		00	01
C	0		1
	1	1	1

$$F2(A,B,C) = \prod (0,2,3,6)$$

		AB	
		00	01
C	0	0	0
	1	0	

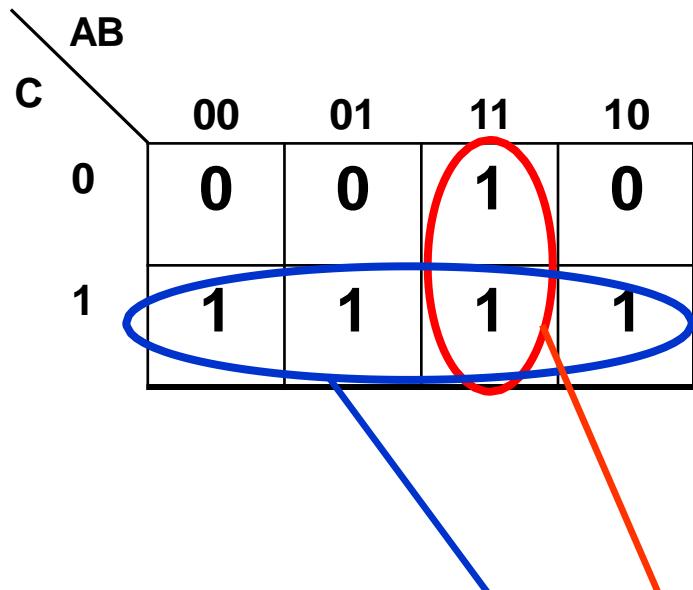
La méthode des tableaux de Karnaugh

□ Simplification d'une fonction

1. Remplir le tableau à partir de la table de vérité ou à partir de la forme canonique.
2. Faire des regroupements : des regroupements de 16,8,4,2,1 cases (Les même termes peuvent participer à plusieurs regroupements).
3. Dans un regroupement :
 - Qui contient un seul terme on ne peut pas éliminer de variables.
 - Qui contient deux termes on peut éliminer une variable (celle qui change d'état).
 - Qui contient 4 termes on peut éliminer 2 variables.
 - Qui contient 8 termes on peut éliminer 3 variables.
 - Qui contient 16 termes on peut éliminer 4 variables.
4. L'expression logique finale est la réunion (la somme) des groupements après simplification et élimination des variables qui changent d'état.

La méthode des tableaux de Karnaugh

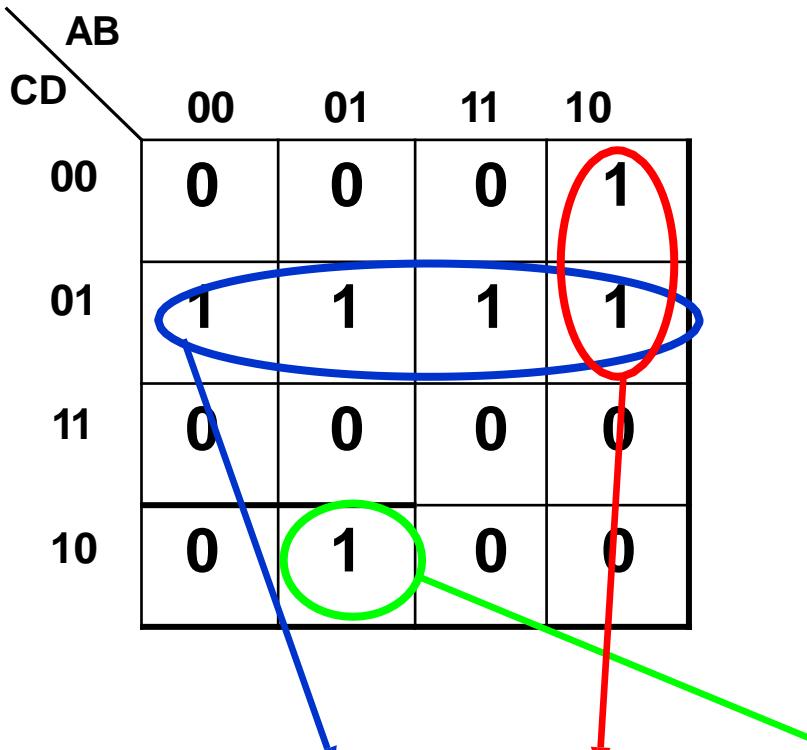
Exemple 1 : 3 variables



$$F(A, B, C) = C + AB$$

La méthode des tableaux de Karnaugh

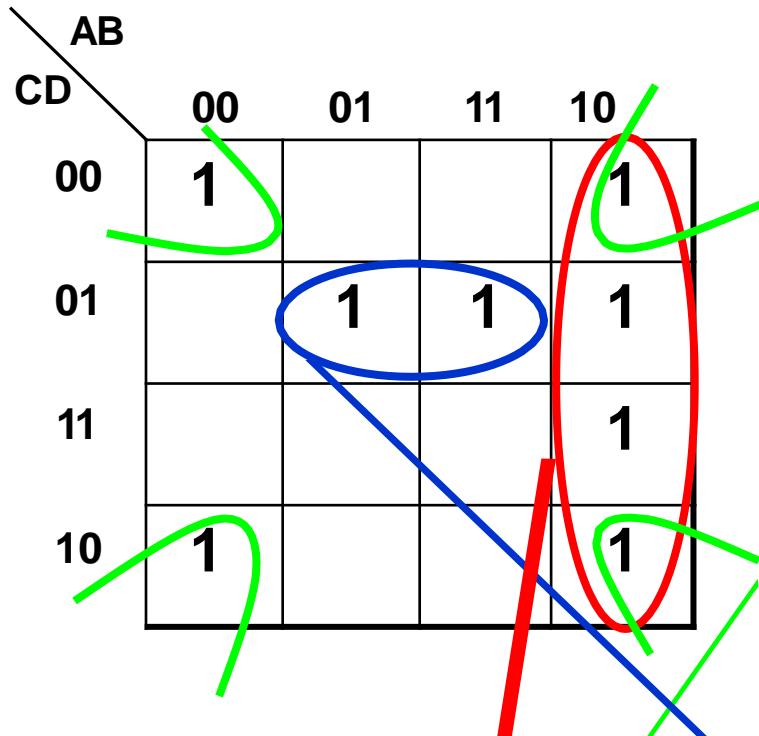
Exemple 2 : 4 variables



$$F(A, B, C, D) = \overline{C} \cdot D + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C \cdot \overline{D}$$

La méthode des tableaux de Karnaugh

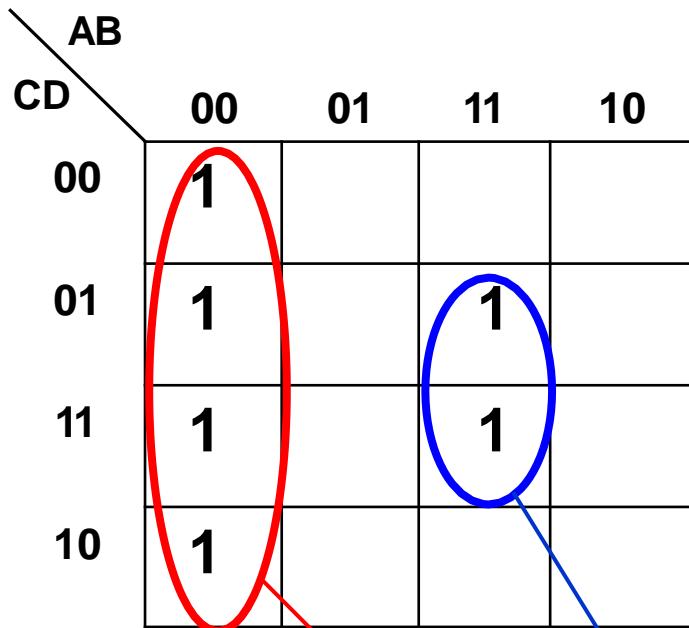
Exemple 3 : 4 variables



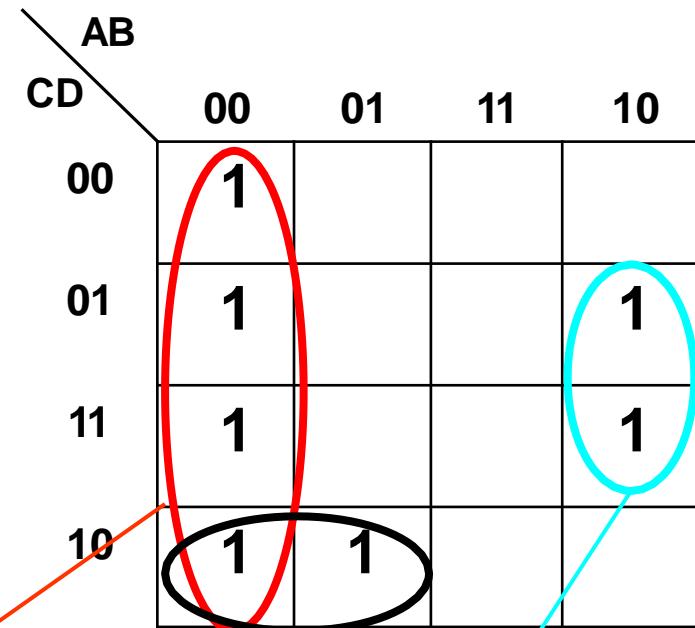
$$F(A, B, C, D) = \overline{AB} + \overline{BD} + B\overline{CD}$$

La méthode des tableaux de Karnaugh

Exemple 4 : 5 variables



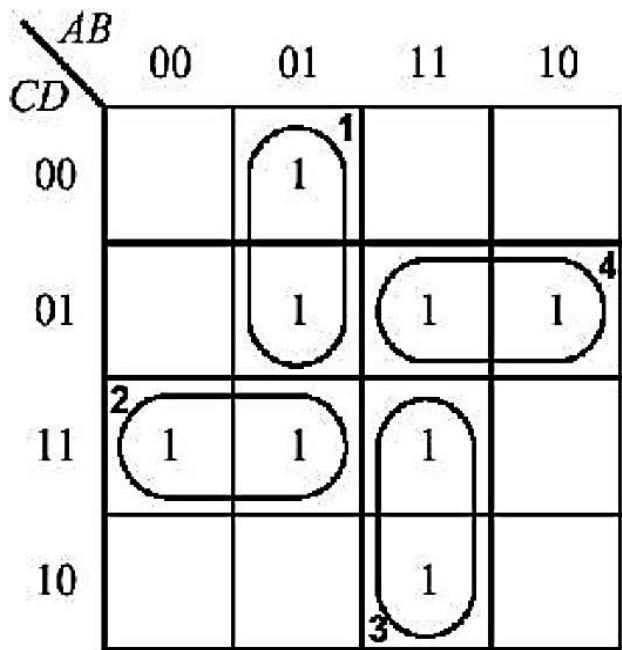
U = 0



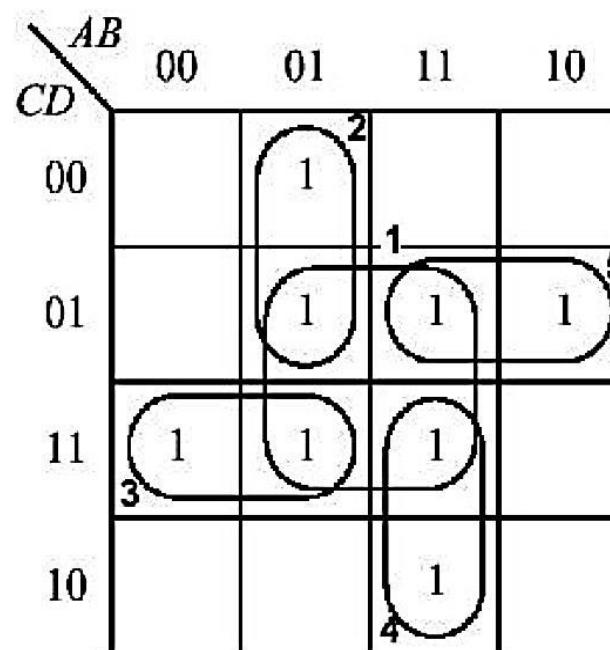
U = 1

$$F(A, B, C, D, U) = \overline{A} \overline{B} + A \cdot B \cdot D \cdot \overline{U} + \overline{A} \cdot C \cdot \overline{D} \cdot U + A \cdot \overline{B} \cdot D \cdot U$$

Karnaugh: Groupement minimal et non minimal



$$F = \overline{A}B\overline{C} + \overline{A}CD + ABC + A'CD$$



$$F = BD + \overline{A}B\overline{C} + \overline{A}CD + ABC + A'\overline{C}D$$

Logique multi-niveaux: tableaux de Karnaugh

On peut généraliser l'algèbre binaire à plus de 2 niveaux

a	b	0	1	Z	X
0	0	X	0	X	
1	X	1	1	X	
Z	0	1	Z	X	
X	X	X	X	X	

0 logique
1 logique
Z déconnecté
X inconnu

Logique multi-niveaux: tableaux de Karnaugh

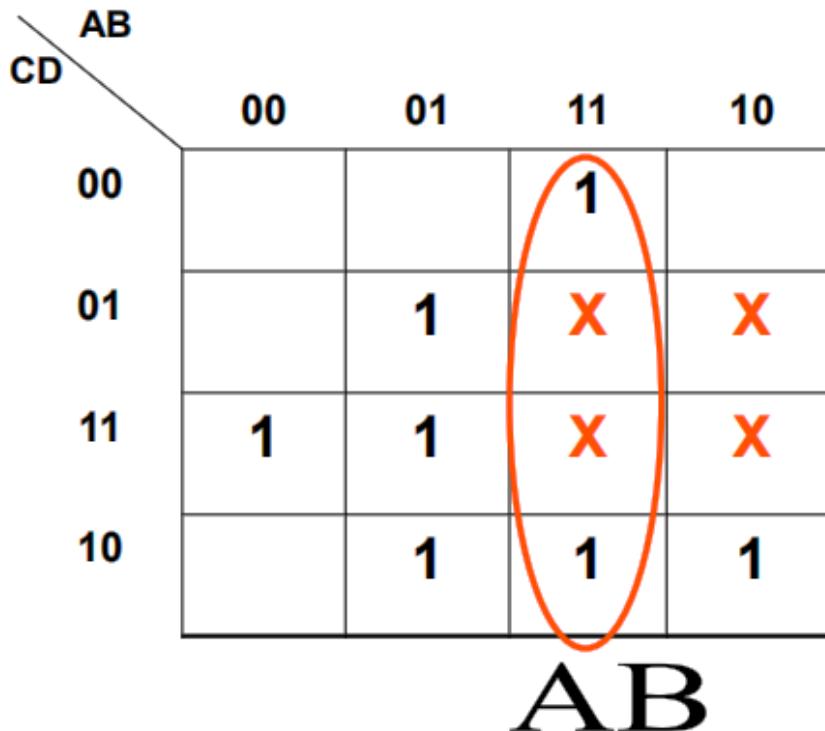
- Pour les cas impossibles ou interdites il faut mettre un **X** dans la T.V .
- Les cas impossibles sont représentées aussi par des **X** dans la table de karnaugh

		AB	CD	00	01	11	10
		CD	00	00		1	
		01	01		1	X	X
		11	11	1	1	X	X
		10	10		1	1	1

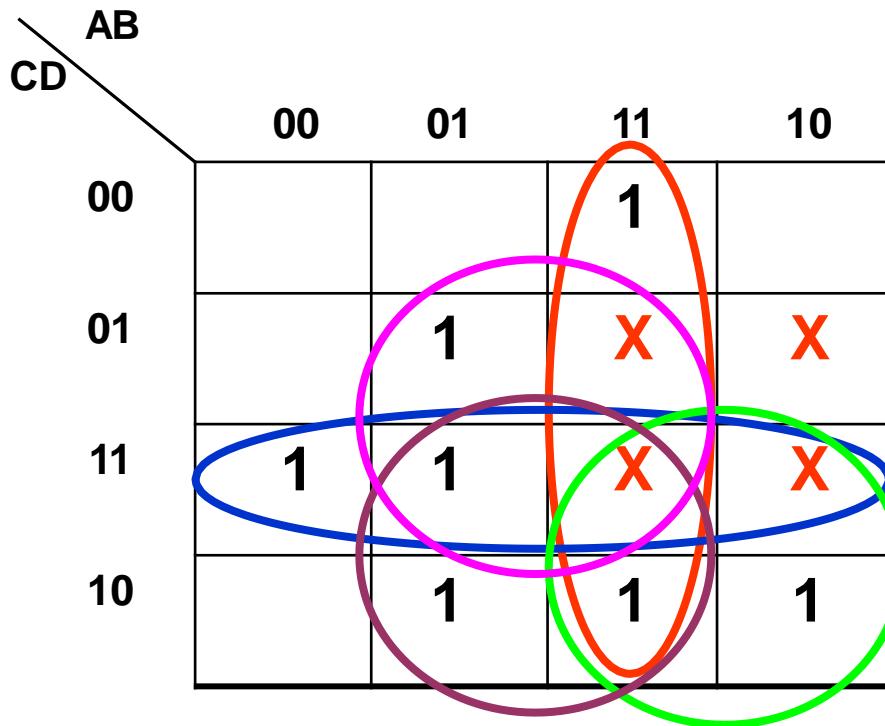
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	X
1	0	1	0	1
1	0	1	1	X
1	1	0	0	1
1	1	0	1	X
1	1	1	0	1
1	1	1	1	X

Logique multi-niveaux: tableaux de Karnaugh

- Il est possible d'utiliser les X dans des regroupements :
 - Soit les prendre comme étant des 1
 - Ou les prendre comme étant des 0
- Il ne faut pas former des regroupement qui contient uniquement des X



Logique multi-niveaux: tableaux de Karnaugh



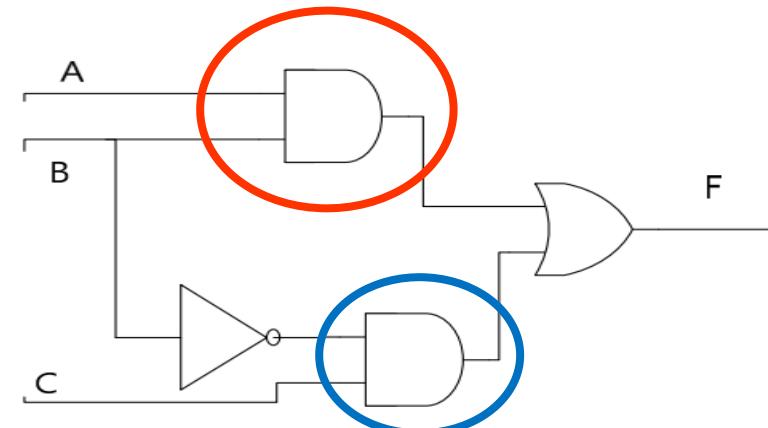
$$AB + CD + BD + AC + BC$$

Schéma d'un circuit logique (Logigramme)

- C'est la traduction de la fonction logique en un schéma électronique.
 - Le principe consiste à remplacer chaque **opérateur logique** par la **porte logique** qui lui correspond.
 - Ordre de priorité des opérateur:
 1. Négation
 2. ET
 3. OU
- Exemple: $x.(\bar{x} + y) + x.y \longrightarrow x.(\bar{x} + y) + (\bar{x}.y)$

Exemple1:

$$F(A, B, C) = A.B + \bar{B}.C$$

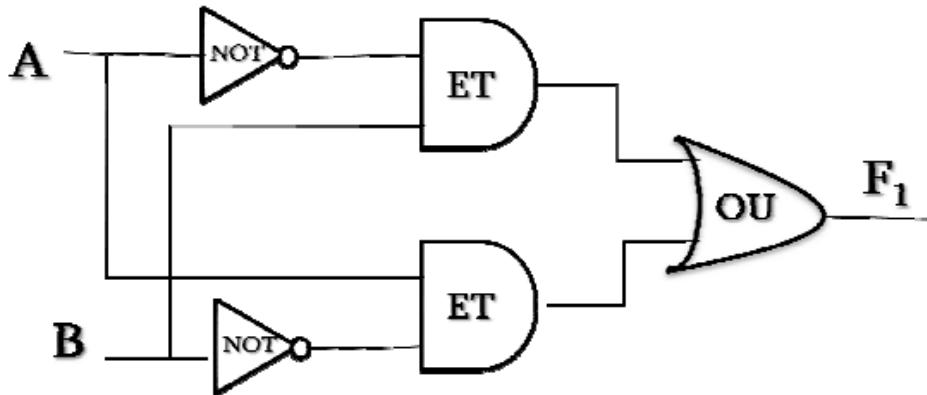


Analyse de circuit logique

A partir du schéma ou du logigramme d'un circuit logique, on peut déduire son fonctionnement en respectant les étapes d'analyse suivantes:

1. Définir la fonction logique à partir du logigramme.
2. Etablir la table de vérité de la fonction logique.
3. Déduire le rôle du circuit.

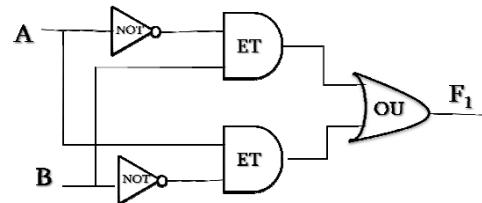
Exemple: Quel est le rôle de ce circuit ?



Analyse de circuit logique

□ Analyse d'un Circuit Logique

Exemple: Quel est le rôle de ce circuit ?



Solution:

$$1. \quad F_1 = \bar{A}B + A\bar{B}$$

2. Table de vérité

A	B	\bar{A}	\bar{B}	$\bar{A}B$	$A\bar{B}$	F_1
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

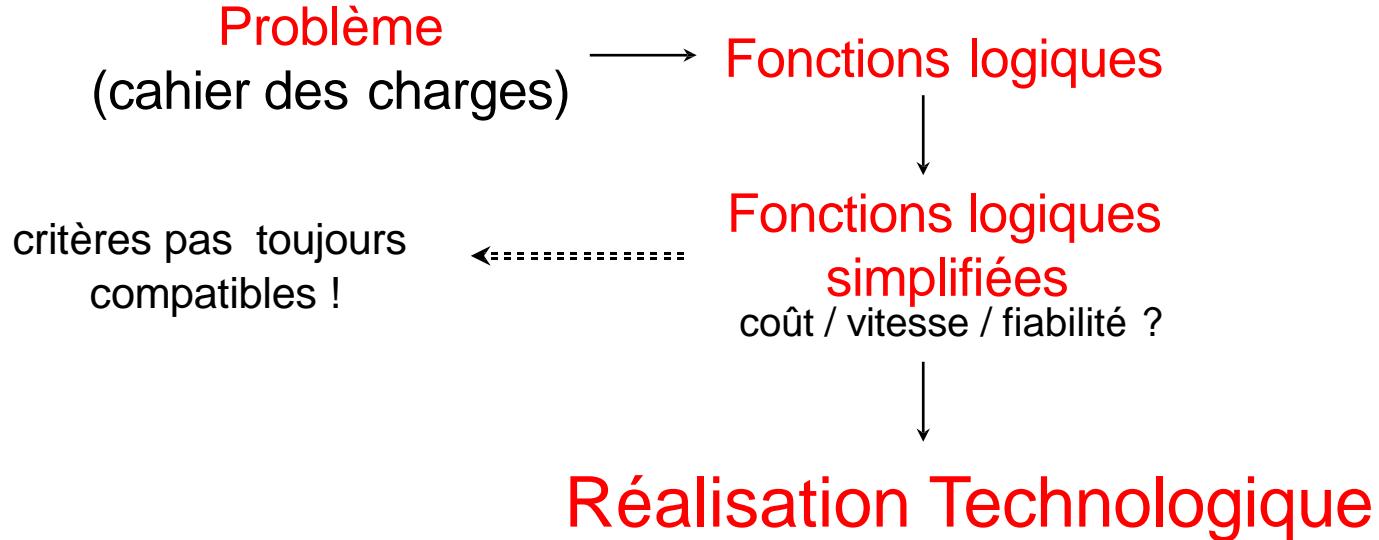
3. Rôle du circuit : $F_1(A, B) = \begin{cases} 1 & \text{si } A \neq B \\ 0 & \text{sinon} \end{cases}$ Test d'Inégalité

Synthèse/Conception d'un Circuit Logique

- Généralement, un circuit est donné sous un format textuel.
- il faut le modéliser en déterminant sa fonction logique à partir de la description textuelle en respectant les étapes suivantes :
 - Définir les variables d'entrée.
 - Définir les variables de sortie.
 - Etablir la table de vérité.
 - Ecrire les équations algébriques des sorties à partir de la table de vérité
 - Effectuer des simplifications (algébrique ou Karnaugh)
 - Dessiner le logigramme du circuit.

Les circuits combinatoires et séquentiels

Introduction

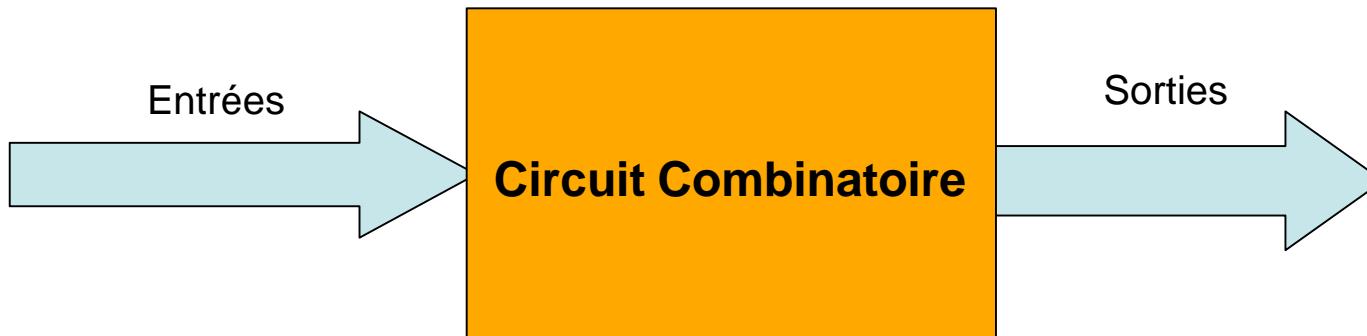


Introduction

- Les circuits logiques sont élaborés à partir de composants **électroniques** – **transistors**
- Types de circuits logiques:
 - **Combinatoires**
 - **Séquentiels**

Circuits combinatoires

- CLC: Circuit Logique Combinatoire
- Support théorique – algèbre de Boole
- Les fonctions de sortie s'expriment selon des expressions logiques des variables d'entrée seulement.
 - Un circuit combinatoire est défini par une ou plusieurs fonctions logiques



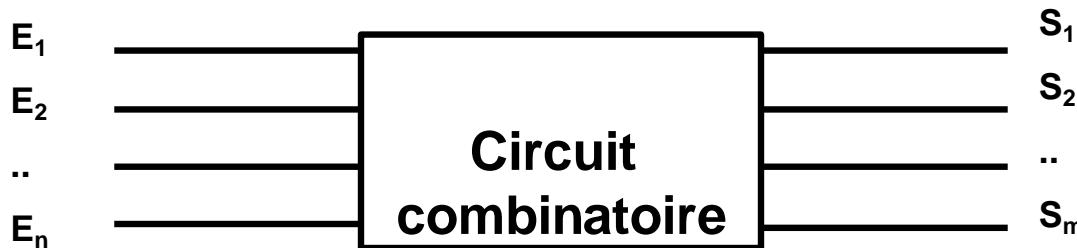
Circuits séquentiels ou à mémoire

- CLS: Circuits Logique Séquentiels
- Support théorique – théorie des automates Finis
- Les fonctions de sortie dépendent non seulement de l'état des variables d'entrée mais également de l'état antérieur de certaines variables de sortie (propriétés de mémorisation)
- **Notion du temps => besoin d'une horloge**



Circuits combinatoires

- Un circuit combinatoire est un circuit numérique dont **les sorties** dépendent uniquement **des entrées**.
- $S_i = F(E_i)$
- $S_i = F(E_1, E_2, \dots, E_n)$

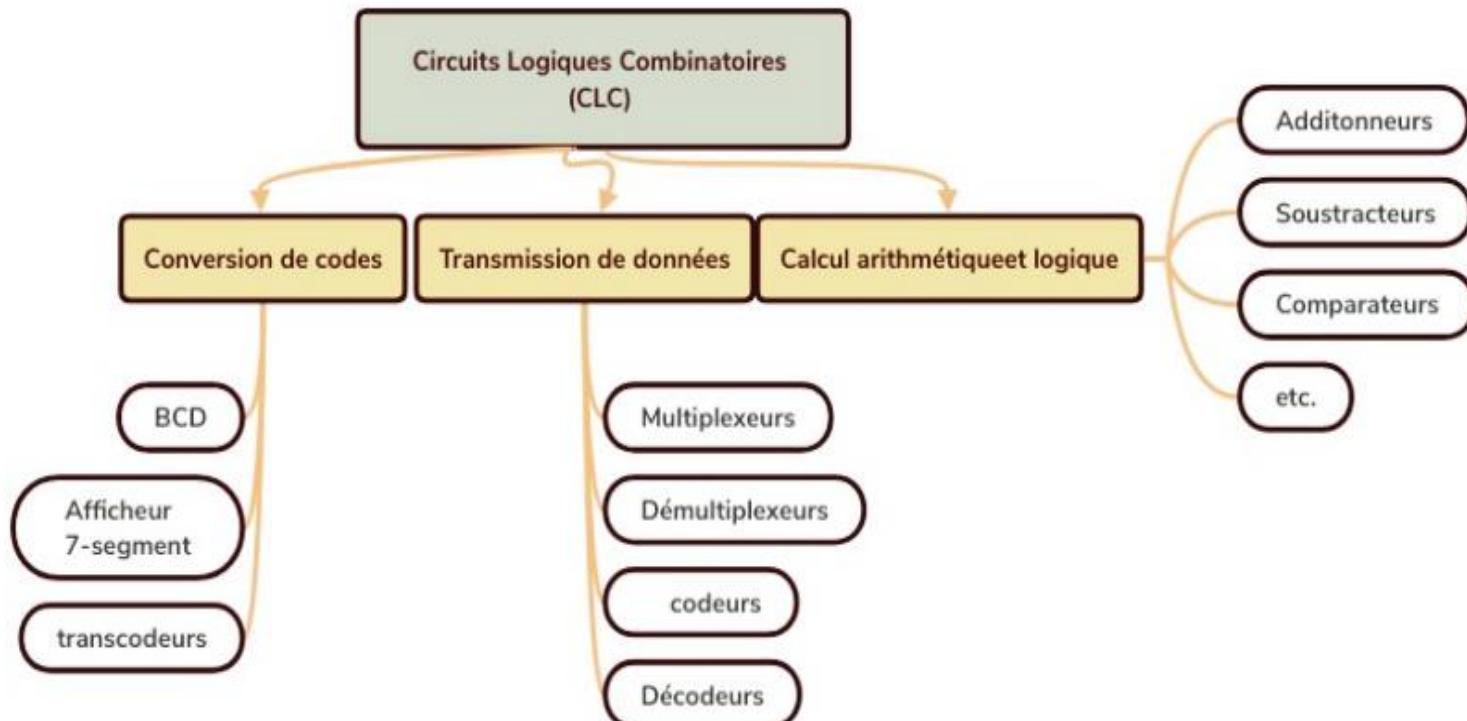


- C'est possible d'utiliser des circuits combinatoires pour réaliser d'autres circuits **plus complexes**.

Circuits combinatoires

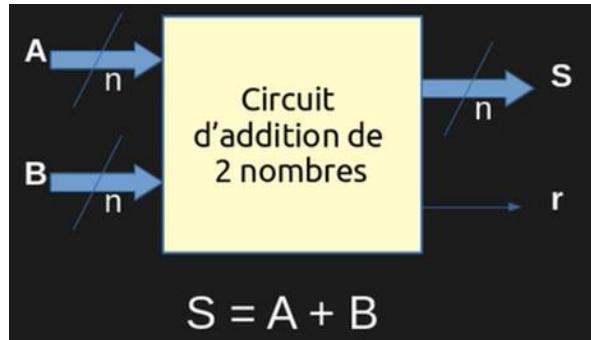
- Le circuit combinatoire est défini lorsque son nombre **d'entrées**, son nombre de **sorties** ainsi que **l'état de chaque sortie** en fonction des entrées ont été précisés
- Ces informations sont fournies grâce à une table de vérité
- La table de vérité d'une fonction de **n variables** a **2^n** lignes - états d'entrée
- Algèbre de Boole et les fonctions logiques sont le support théorique des circuits combinatoires

Classification des CLC



Additionneur

- **Problème:** Faire la synthèse d'un circuit d'addition de 2 nombres A et B sur n bits chacun

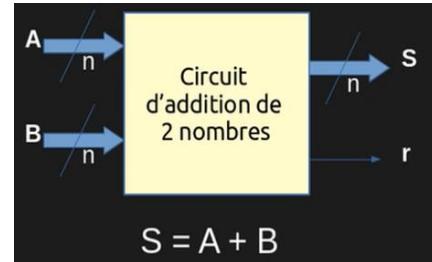


- On a:
 - $2n$ entrées
 - $n+1$ sorties au moins
- Le lien entre les sorties et les entrées est déduit en utilisant les règles d'addition

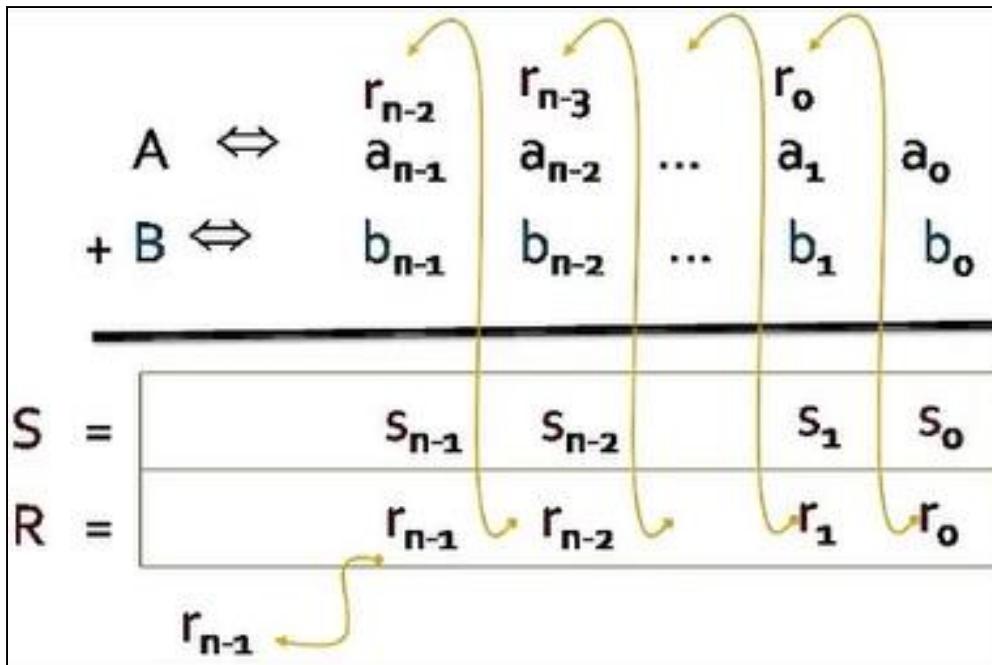
$0+0 = 0$	retenue 0
$0+1 = 1 + 0 = 1$	retenue 0
$1 + 1 = 0$	retenue 1
$1+1+1 = 1$	retenue 1

Additionneur

- Les règles s'appliquent pour chaque niveau des bits des deux nombres **A** et **B** en allant du bits de poids faible vers le bits de poids fort.

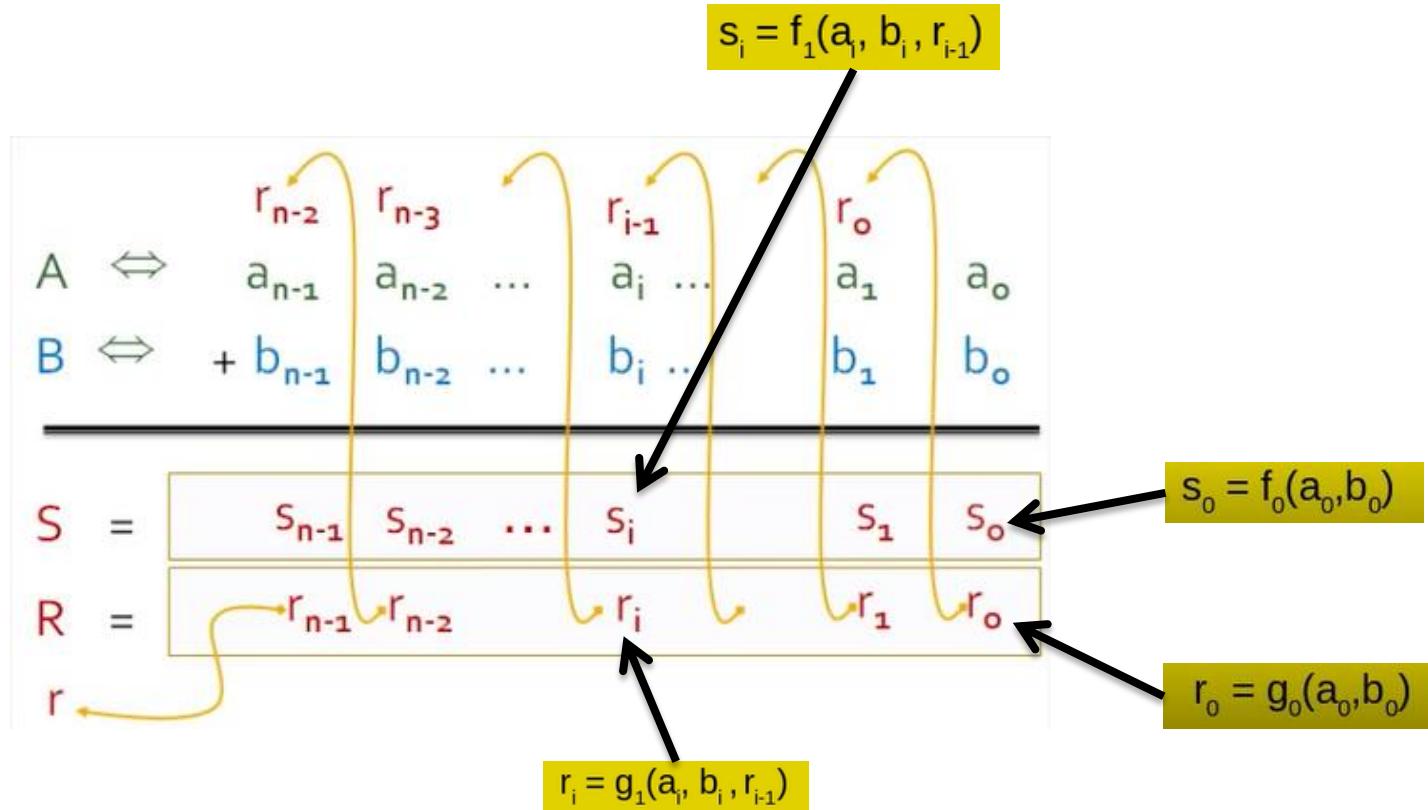


$0+0 = 0$	retenue 0
$0+1 = 1 + 0 = 1$	retenue 0
$1+1 = 0$	retenue 1
$1+1+1 = 1$	retenue 1



Additionneur

- Les sorties dépendent des entrées de l'étage d'addition



Additionneur

- Il suffit donc de trouver 4 fonctions:

$$\begin{array}{l} \bullet \quad s_0 = f_0(a_0, b_0) \\ \bullet \quad r_0 = g_0(a_0, b_0) \end{array} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Etage 0}$$

$$\begin{array}{l} \bullet \quad s_i = f_i(a_i, b_i, r_{i-1}) \\ \bullet \quad r_i = g_i(a_i, b_i, r_{i-1}) \end{array} \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Etage } i$$

Additionneur

- Trouver les 2 fonctions de l'étage 0:

- $s_0 = f_0(a_0, b_0)$
- $r_0 = g_0(a_0, b_0)$

a_0	b_0	s_0	r_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{aligned}s_0 &= m_1 + m_2 \\ r_0 &= m_3\end{aligned}$$

$$S_0 = \bar{a}_0 b_0 + a_0 \bar{b}_0$$



$$\begin{aligned}S_0 &= a_0 \oplus b_0 \\ r_0 &= a_0 b_0\end{aligned}$$

Additionneur

- Trouver les 2 fonctions de l'étage i:

- $s_i = f_i(a_i, b_i, r_{i-1})$

- $r_i = g_i(a_i, b_i, r_{i-1})$

a_i	b_i	r_{i-1}	s_i	r_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = m_1 + m_2 + m_4 + m_7$$

$$R_i = m_3 + m_5 + m_6 + m_7$$

Additionneur

- Simplification de la fonction S_i :

$$\begin{aligned} s_i &= m_1 + m_2 + m_4 + m_7 \\ &= \bar{a}_i \bar{b}_i r_{i-1} + \bar{a}_i b_i \bar{r}_{i-1} + \\ &\quad a_i \bar{b}_i \bar{r}_{i-1} + a_i b_i r_{i-1} + \\ &= \bar{a}_i (\bar{b}_i r_{i-1} + b_i \bar{r}_{i-1}) + \\ &\quad a_i (\bar{b}_i \bar{r}_{i-1} + b_i r_{i-1}) \\ &= \bar{a}_i (b_i \oplus r_{i-1}) + a_i (\bar{b}_i \oplus \bar{r}_{i-1}) \\ &= a_i \oplus (b_i \oplus r_{i-1}) \end{aligned}$$

Additionneur

- Simplification de la fonction r_i :

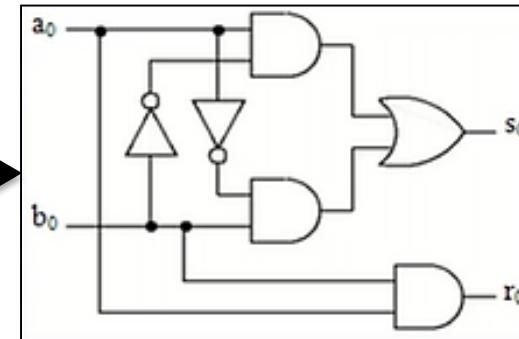
$$\begin{aligned}r_i &= m_3 + m_5 + m_6 + m_7 \\&= \bar{q}_i b_i r_{i-1} + q_i \sqrt{b_i} r_{i-1} + \\&\quad q_i b_i \bar{r}_{i-1} + q_i b_i r_{i-1} \\&= \bar{q}_i b_i r_{i-1} + q_i \sqrt{b_i} r_{i-1} + q_i b_i (\bar{r}_{i-1} + r_{i-1}) \\&= \bar{q}_i b_i r_{i-1} + q_i \sqrt{b_i} r_{i-1} + q_i b_i \\&= (\bar{q}_i b_i + q_i \sqrt{b_i}) r_{i-1} + q_i b_i \\&= (q_i \oplus b_i) r_{i-1} + q_i b_i\end{aligned}$$

Additionneur

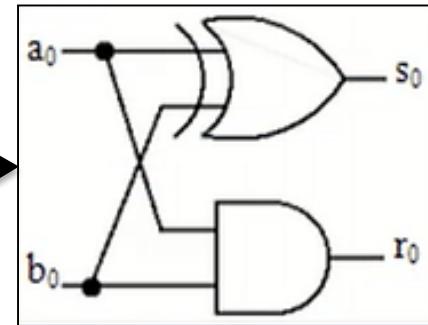
- Logigramme de s_0 et r_0 : (Demi Additionneur)

(Demi Additionneur) => Le premier étage d'un additionneur à 1 bits (les deux premier bits)

$$\begin{aligned}S_0 &= \bar{a}_0 b_0 + a_0 \bar{b}_0 \\r_0 &= a_0 b_0\end{aligned}$$



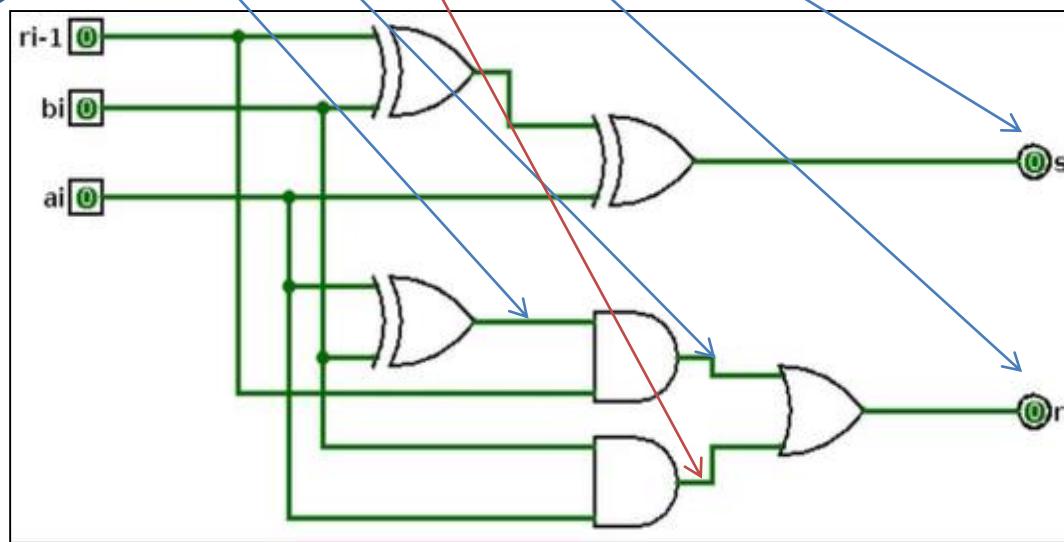
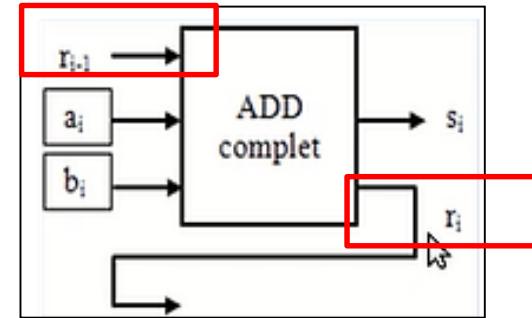
$$\begin{aligned}S_0 &= a_0 \oplus b_0 \\r_0 &= a_0 b_0\end{aligned}$$



Additionneur(Additionneur complet)

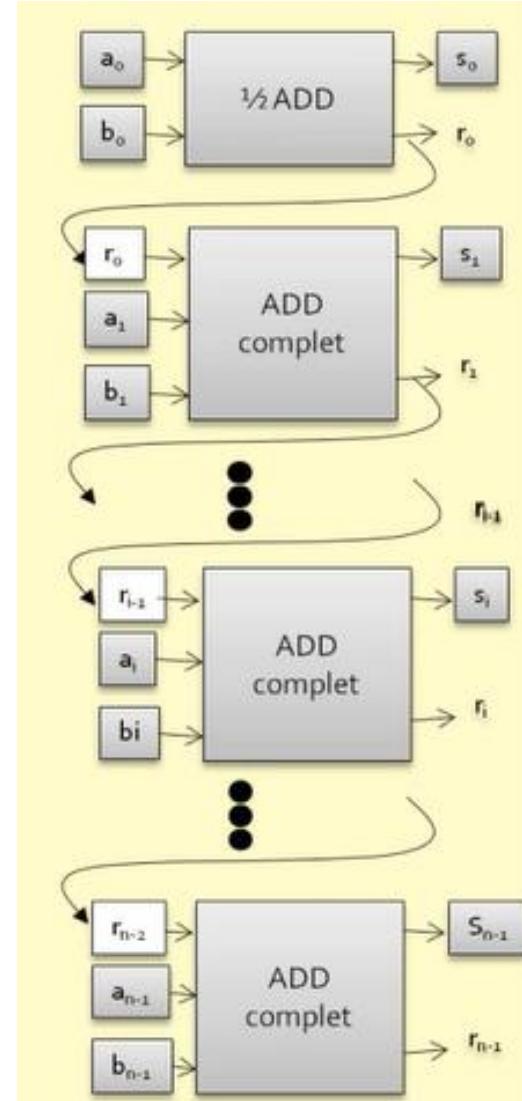
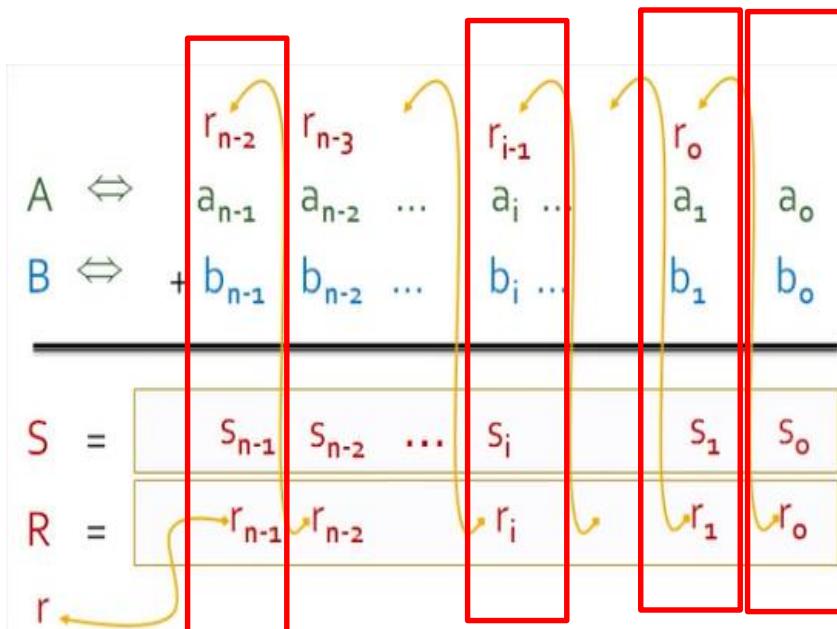
- Logigramme de S_i et r_i :

$$S_i = a_i \oplus (b_i \oplus r_{i-1})$$
$$r_i = (a_i \oplus b_i) r_{i-1} + (a_i b_i)$$



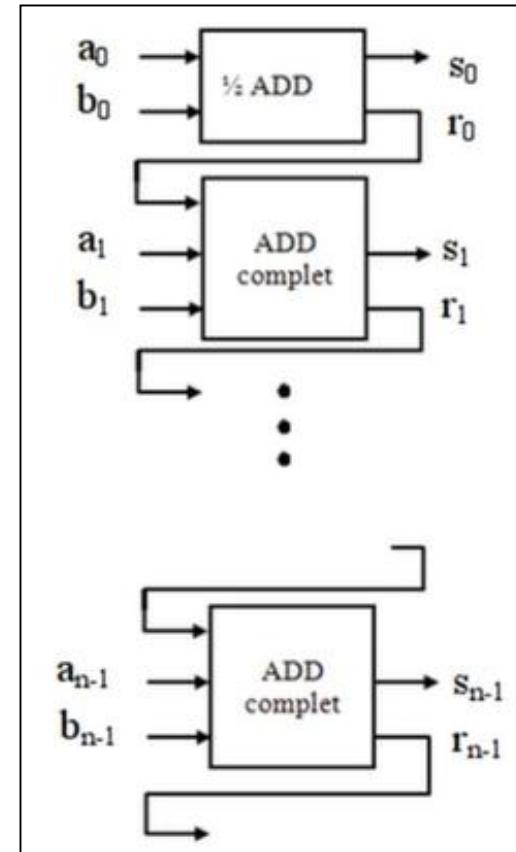
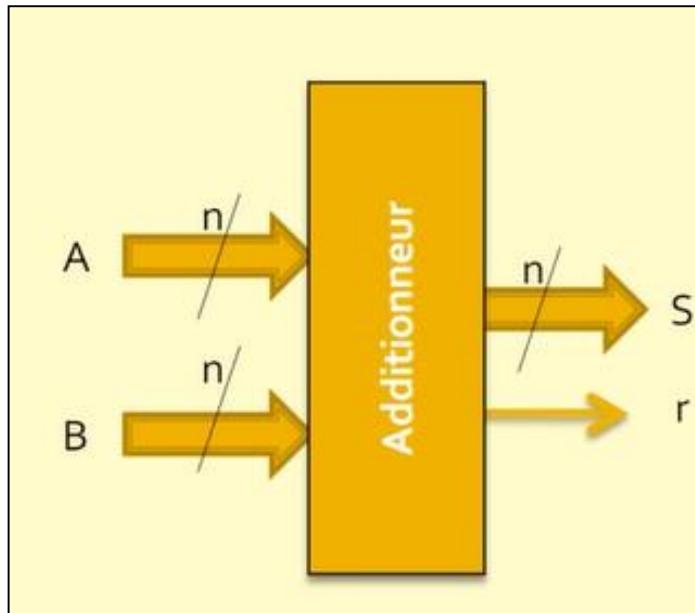
Additionneur

- Logigramme d'un circuit additionneur:



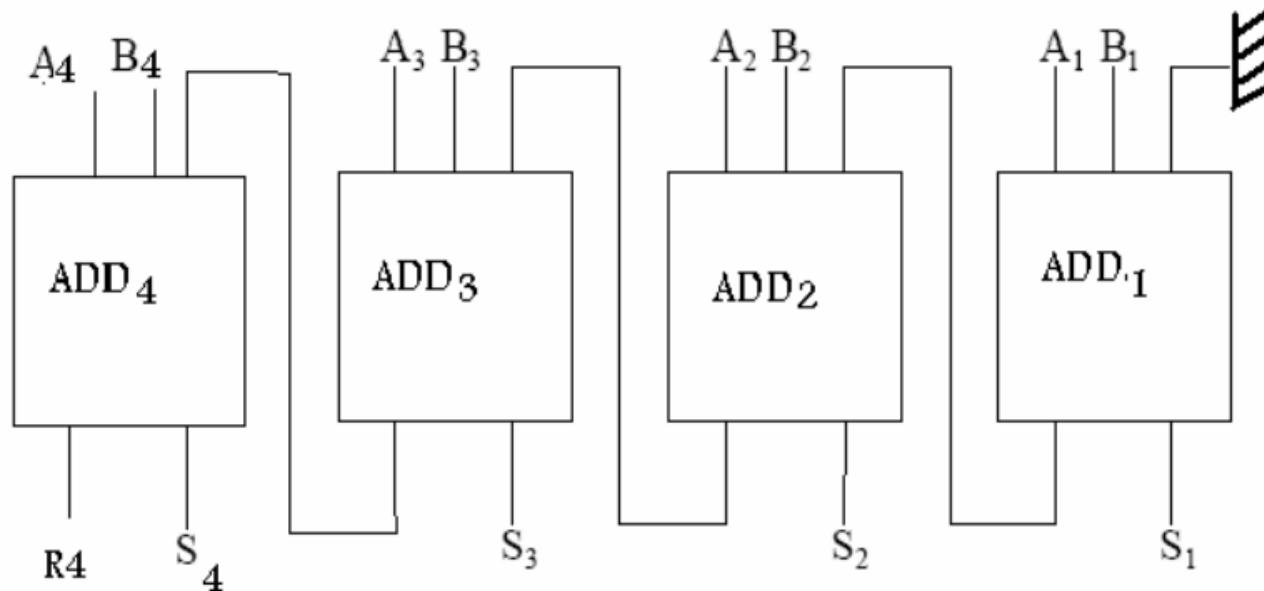
Additionneur

- Schéma d'un additionneur n bits à propagation de la retenue



Additionneur

- Schéma d'un additionneur 4 bits



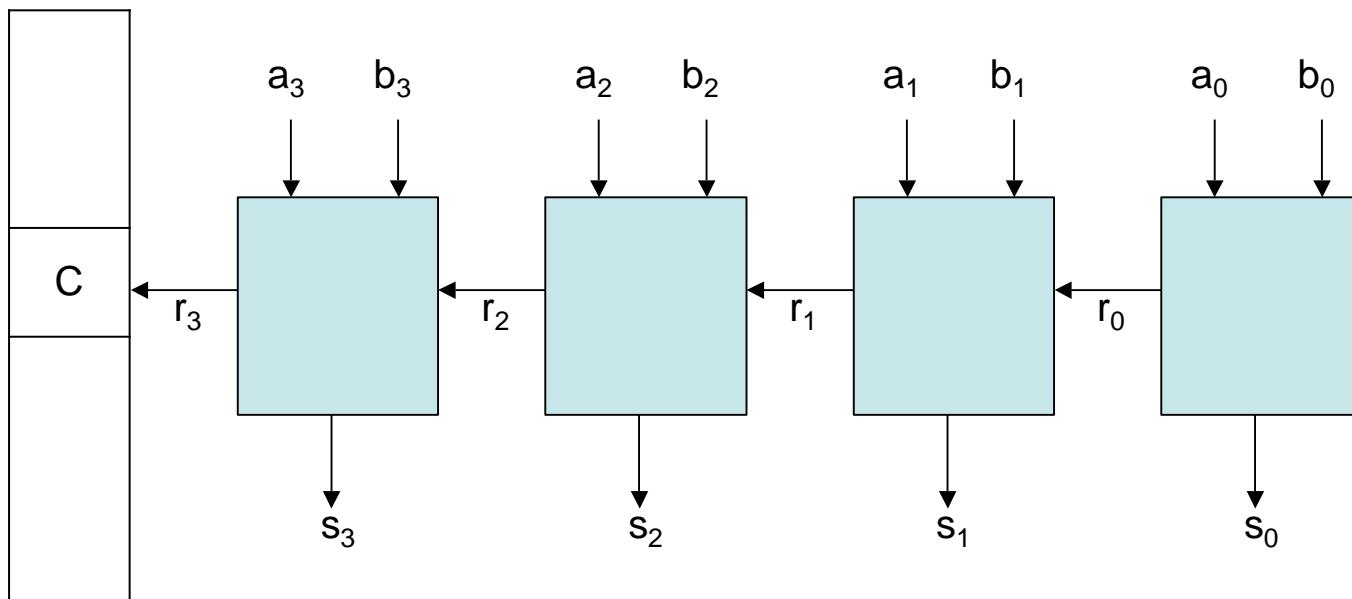
Additionneur

- Indicateur de Retenu (carry)
 - Lors d'une opération arithmétique effectuée sur des nombres de n bits, un bit de retenu peut être généré.
 - Ce bit de retenu mémorisé par l'indicateur C du registre d'état du processeur, correspond au niveau de l'additionneur n bits, à une retenue r_{n-1} égale à 1 pour l'additionneur complet 1 bit.

Additionneur

- Indicateur de Retenu pour l'additionneur 4 bits

Registre d'état



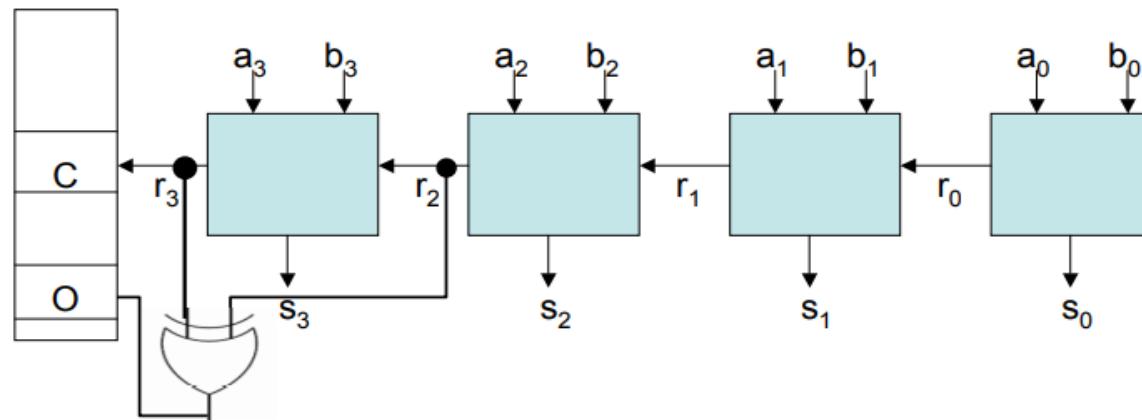
Additionneur

- Indicateur de débordement (overflow)
 - Lors d'une opération arithmétique mettant en jeu des nombres de n bits et de **même signe**, le résultat peut être en dehors de l'intervalle des nombres représentables sur n bits par la convention choisie pour la représentation de ces nombres signés.
 - Dépassement de capacité ne peut se produire que lors de l'addition de 2 nombres de même signe

Additionneur

- Indicateur de débordement (overflow)
 - Overflow peut être détecté en effectuant un test de comparaison entre r_{n-2} et r_{n-1}
 - $\text{Overflow} \Leftrightarrow r_{n-2} \neq r_{n-1}$

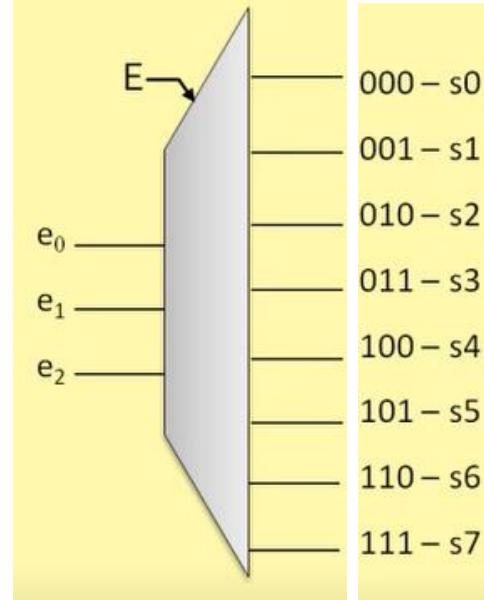
Registre d'état



$$O = r_{n-2} \oplus r_{n-1}$$

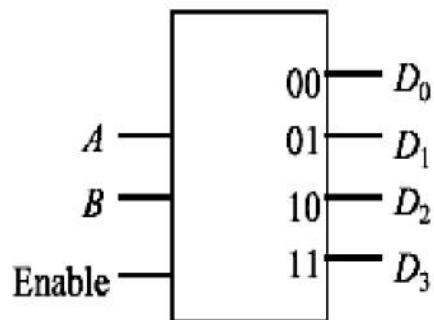
Décodeur

- **Un circuit qui dispose :**
 - n entrées
 - 2^n sorties
 - Validation (Enable)
- Il permet d'activer une ligne de sortie (sélection) correspondante à la configuration présentée en entrée.
- **Principe de fonctionnement :**
 - Seule la sortie dont le numéro est correspond au **Minterme** composé des variables d'entrées sera à 1.
 - Si l'entrée de validation à 0, aucun sortie n'est à 1
 - => On peut déduire directement **les fonctions de sorties**
- On écrit : $S_i = E \cdot m_i$



Décodeur

- Exemple:



		Enable = 1			
<i>A</i>	<i>B</i>	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

		Enable = 0			
<i>A</i>	<i>B</i>	D_0	D_1	D_2	D_3
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

$$D_0 = \overline{A} \overline{B}$$

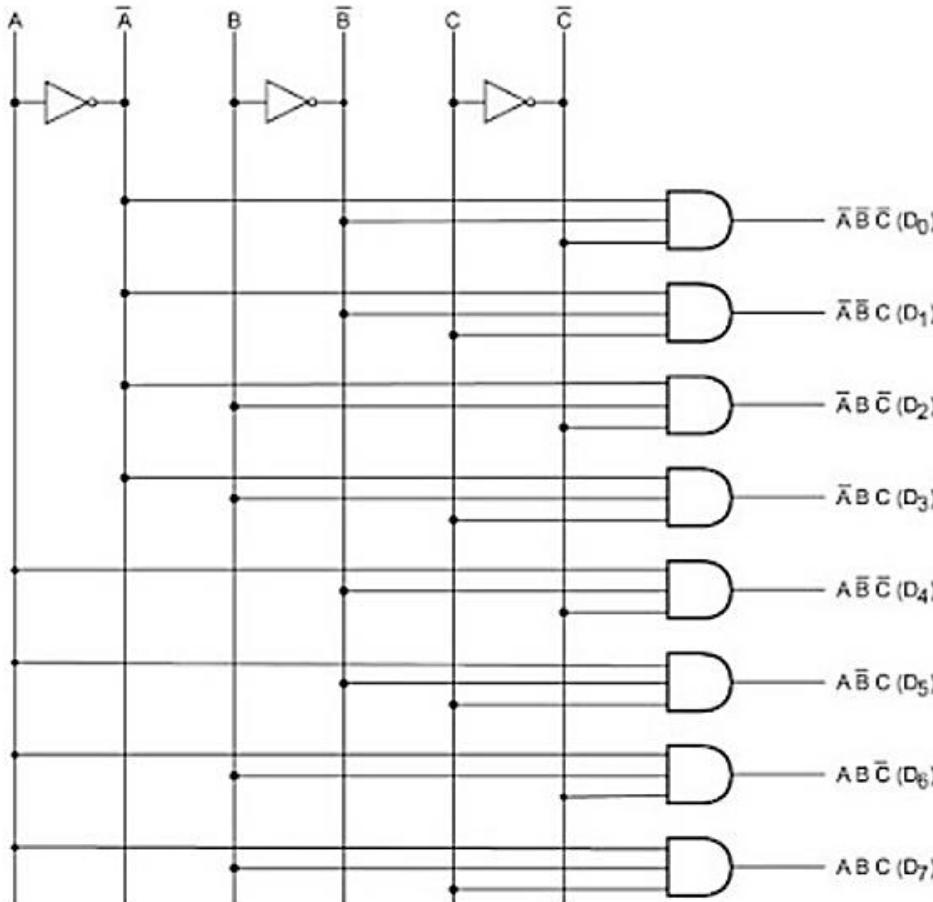
$$D_1 = \overline{A} B$$

$$D_2 = A \overline{B}$$

$$D_3 = A B$$

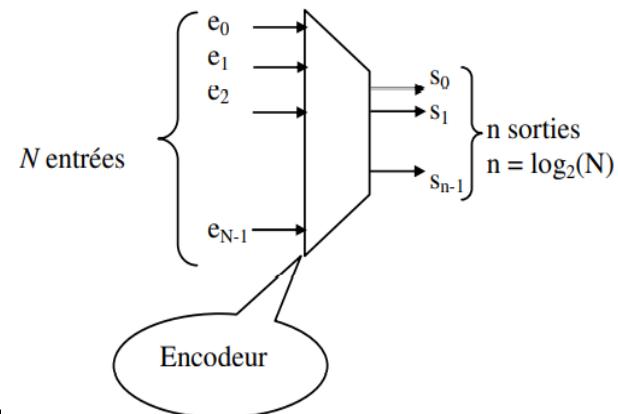
Décodeur

- Exemple (3 entrées):



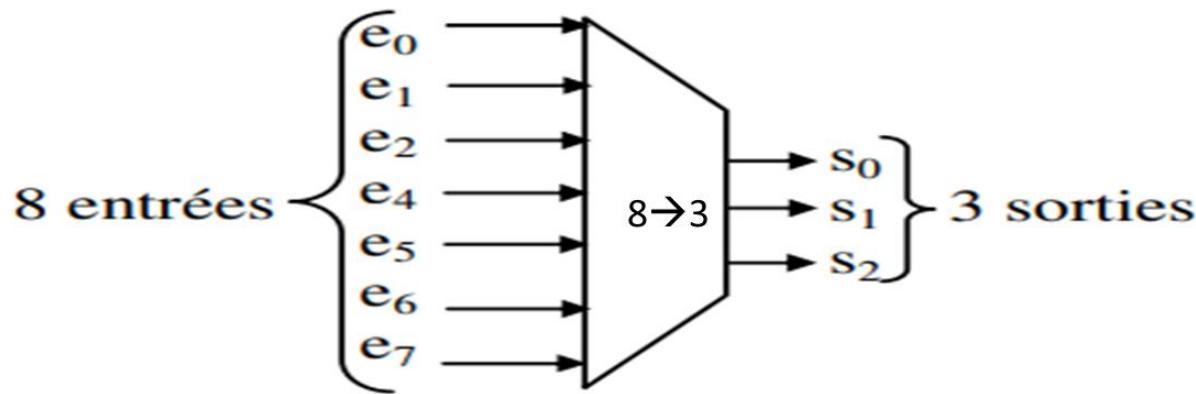
Encodeur

- L'encodeur est un circuit qui réalise la fonction inverse du décodage.
- Un circuit qui dispose :
 - 2^n entrées
 - n sorties
 - Validation (Enable)
- **Principe de fonctionnement :**
 - On suppose que seule une entrée peut être à 1 à un moment donné.
 - Toutes les entrées sont numérotées de 0 à $N-1$.
 - A chaque fois qu'une entrée est à 1, on obtient en sortie le code binaire qui correspond au numéro de l'entrée mise à 1.

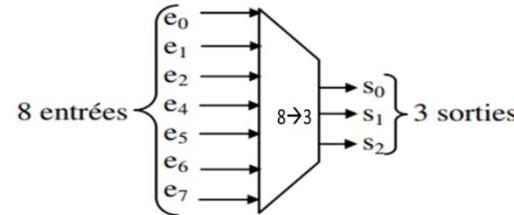


Encodeur

- Exemple: Codeur $8 \rightarrow 3$



Encodeur



Entrées	s_2	s_1	s_0	L'entrée indiquée par les sorties
$e_0=1$	0	0	0	l'entrée 0 (e_0)
$e_1=1$	0	0	1	l'entrée 1 (e_1)
$e_2=1$	0	1	0	l'entrée 2 (e_2)
$e_3=1$	0	1	1	l'entrée 3 (e_3)
$e_4=1$	1	0	0	l'entrée 4 (e_4)
$e_5=1$	1	0	1	l'entrée 5 (e_5)
$e_6=1$	1	1	0	l'entrée 6 (e_6)
$e_7=1$	1	1	1	l'entrée 7 (e_7)

On suppose que seule une entrée peut être à 1 à un moment donné.

Encodeur

- **Equations :**

La sortie s_0 est à 1 lorsque le numéro de l'entrée activée est impair (e_1 OU e_3 OU e_5 OU e_7).

De ce fait : $s_0 = e_1 + e_3 + e_5 + e_7$

La sortie s_1 est à 1 lorsque le numéro de l'entrée activée est 2 OU 3 OU 6 OU 7.

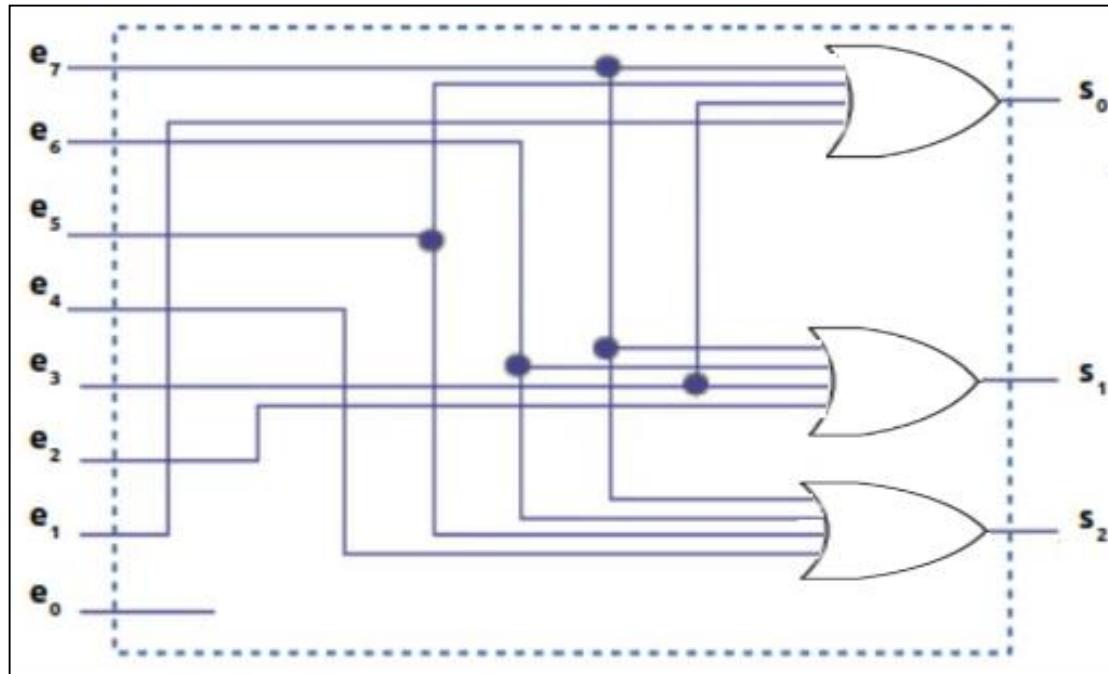
De ce fait : $s_1 = e_2 + e_3 + e_6 + e_7$

La sortie s_2 est à 1 lorsque le numéro de l'entrée activée est 4 OU 5 OU 6 OU 7 .

De ce fait : $s_2 = e_4 + e_5 + e_6 + e_7$

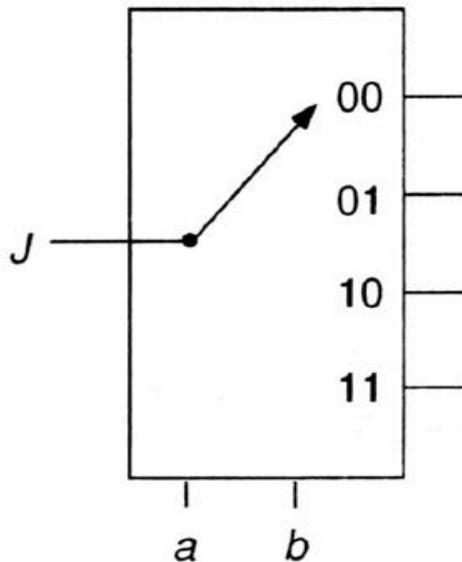
Encodeur

- Ce qui nous donne le logigramme suivant:



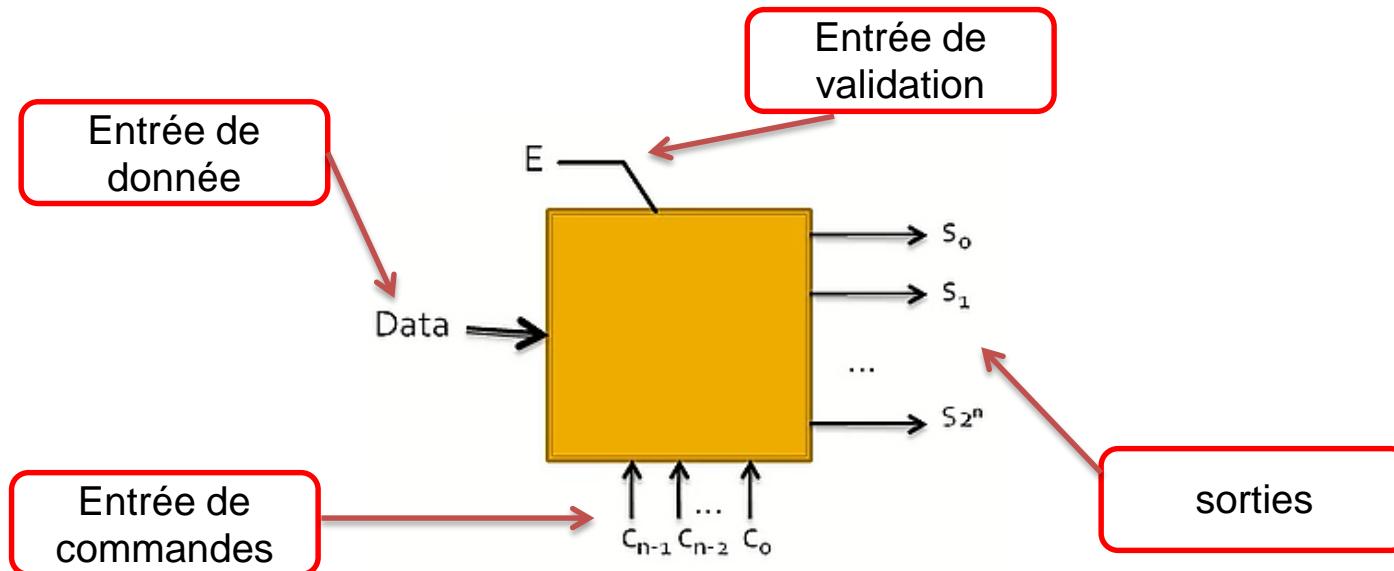
Démultiplexeur

- Un circuit avec **une entrée** et **n sorties** et qui met en relation cette entrée avec une sortie et une seule.
- Un DeMux est une sorte d'aiguilleur d'une entrée vers une sortie parmi 2^n selon le numéro indiqué par les entrées de commandes.



Démultiplexeur

- Une entrée de donnée: Data
- N entrées de commandes: $C_{n-1}, C_{n-2} \dots, C_0$
- Une entrée de validation: E
- $N = 2^n$ sorties : S_{2^n}, \dots, S_0

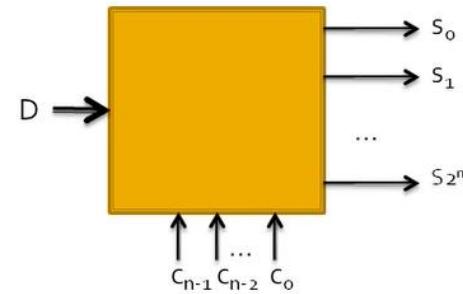


Démultiplexeur

- Fonctionnement:**

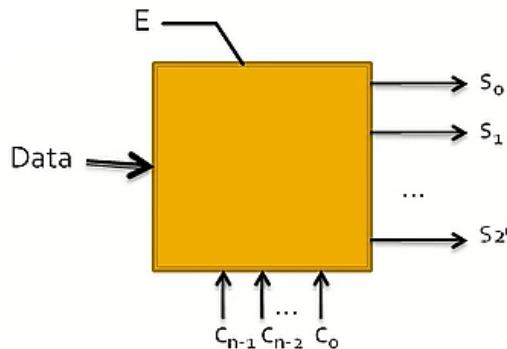
La sortie S_i est égale à l'entrée de donnée si le Minterme formé par les entrées de commande ($C_{n-1}, C_{n-2}, \dots, C_0$) est égale à i

$$S_i = m_i \cdot D$$



- Si on rajoute une entrée de validation la sortie S_i devient:

$$S_i = E \cdot m_i \cdot D$$



Démultiplexeur

- Exemple:
 - DEMUX à 3 entrées de commandes et une entrée de données D

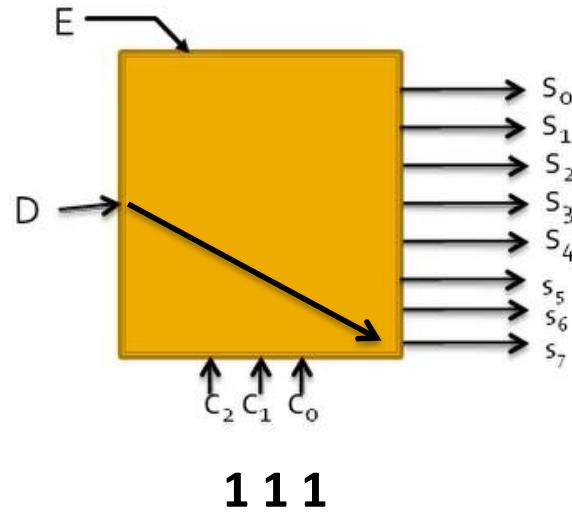
$$s_0 = E \cdot D \cdot m_0 = E \cdot D \cdot (\bar{c}_2 \cdot \bar{c}_1 \cdot \bar{c}_0)$$

$$s_1 = E \cdot D \cdot m_1 = E \cdot D \cdot (\bar{c}_2 \cdot \bar{c}_1 \cdot c_0)$$

$$s_2 = E \cdot D \cdot m_2 = E \cdot D \cdot (\bar{c}_2 \cdot c_1 \cdot \bar{c}_0)$$

...

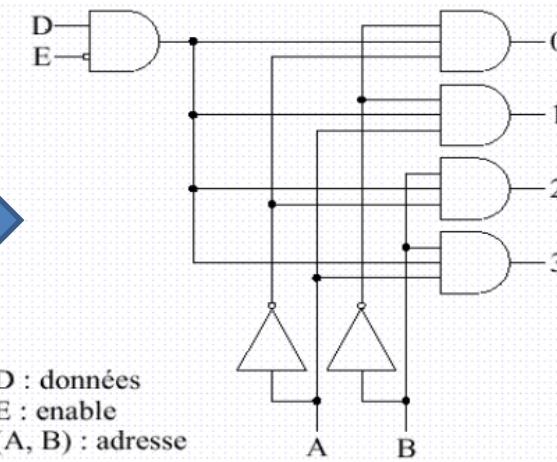
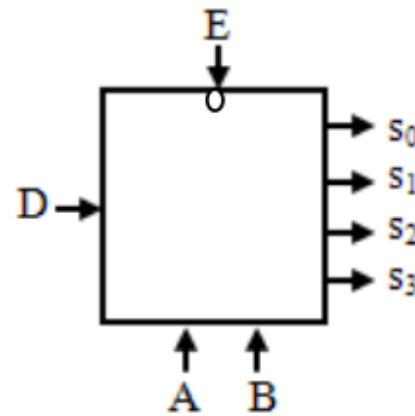
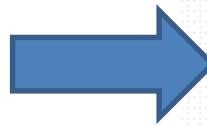
$$s_7 = E \cdot D \cdot m_7 = E \cdot D \cdot (c_2 \cdot c_1 \cdot c_0)$$



Démultiplexeur

- Exemple:
- Table de vérité:

E	B	A	S ₀	S ₁	S ₂	S ₃	Produit
0	0	0	D	0	0	0	$\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D$
0	0	1	0	D	0	0	$\bar{A} \cdot \bar{B} \cdot C \cdot D$
0	1	0	0	0	D	0	$\bar{A} \cdot B \cdot \bar{C} \cdot D$
0	1	1	0	0	0	D	$\bar{A} \cdot B \cdot C \cdot D$
1	0	0	0	0	0	0	
1	0	1	0	0	0	0	
1	1	0	0	0	0	0	
1	1	1	0	0	0	0	

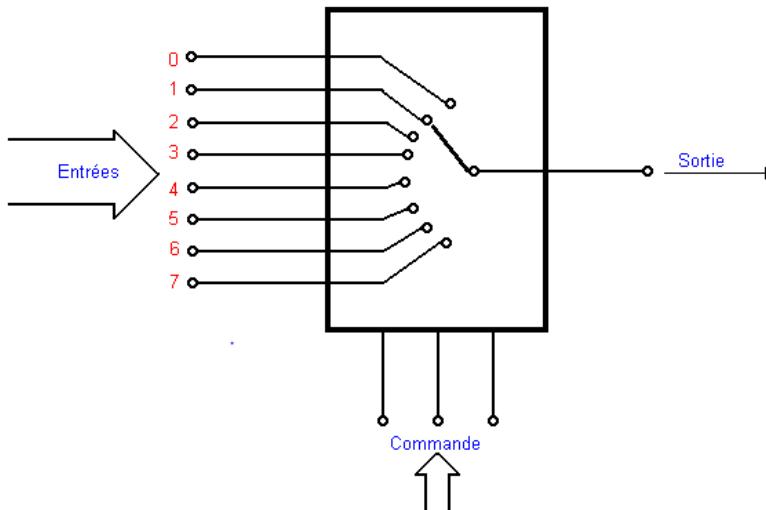


D : données
E : enable
(A, B) : adresse

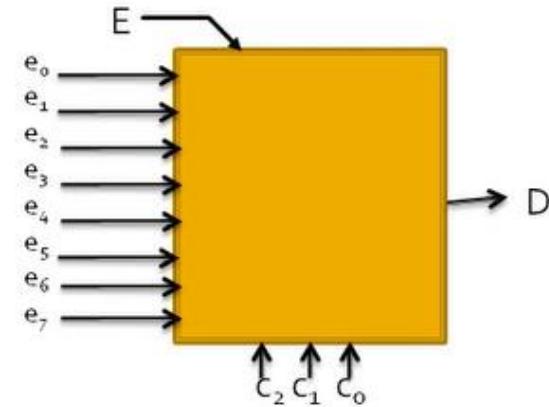
entrée de validation E activée à 0

Multiplexeur

- Le multiplexeur est un circuit combinatoire sélecteur qui possède **2^n entrées** d'information, **n entrées de commande**, **une entrée de validation** et une **seule sortie**.
- Son rôle consiste à sélectionner, à l'aide de signaux de commande, une des entrées et à la lier à la sortie.

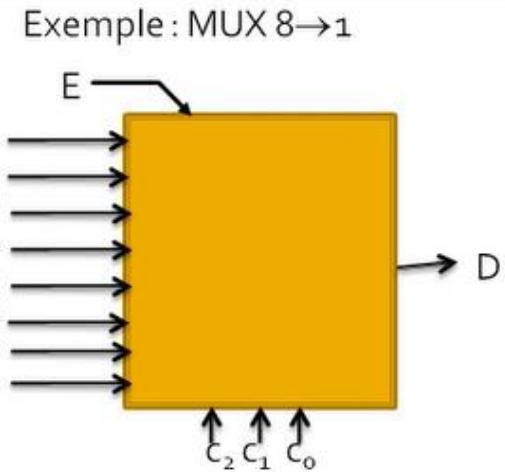
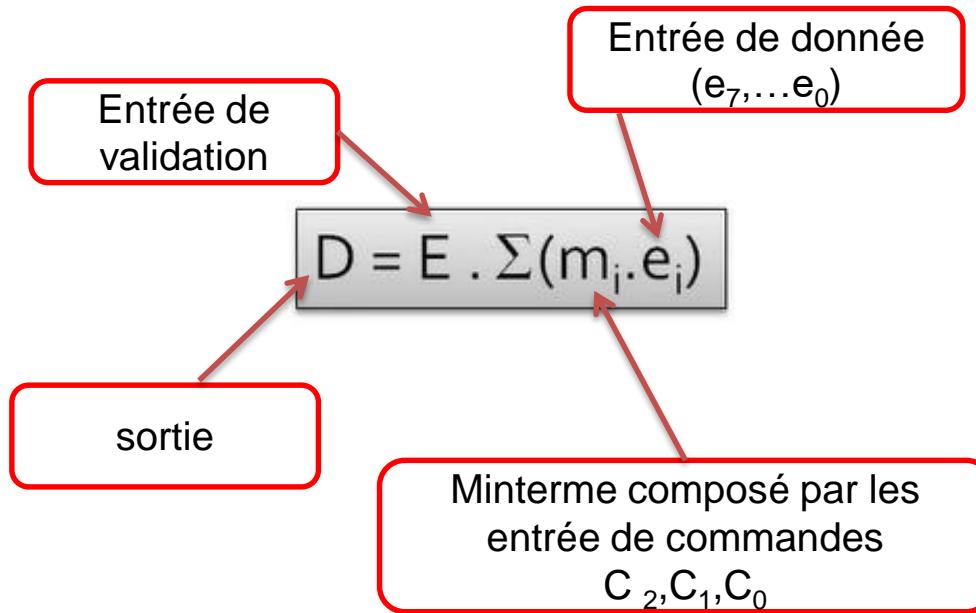


Exemple : MUX 8 \rightarrow 1



Multiplexeur

- Le multiplexeur est un circuit combinatoire sélecteur qui possède **2ⁿ entrées** d'information, **n entrées de commande**, **une entrée de validation** et une **seule sortie**.
- Son rôle consiste à sélectionner, à l'aide de signaux de commande, une des entrées et à la lier à la sortie.

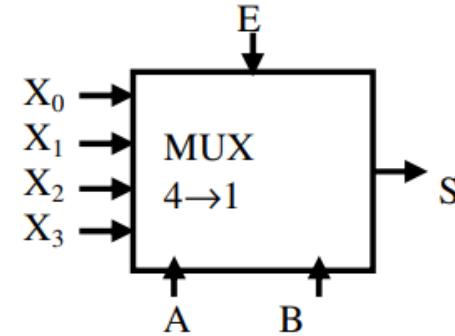


Multiplexeur

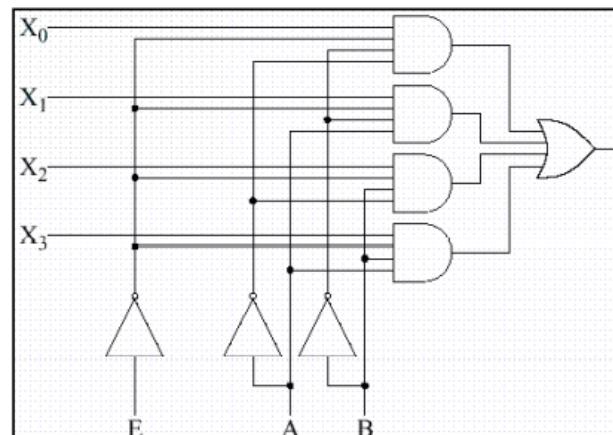
- **Exemple:** un multiplexeur à 4 entrées, deux lignes de commande, et une ligne de validation.

Table de vérité

E	B	A	S
0	0	0	X_0
0	0	1	X_1
0	1	0	X_2
0	1	1	X_3
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



$$S = \bar{E}\bar{B}\bar{A}X_0 + \bar{E}\bar{B}AX_1 + \bar{E}B\bar{A}X_2 + \bar{E}BA\bar{X}_1$$



entrée de validation E activée à 0

Générateur de fonction

- Les circuits comme les **décodeurs** et les **multiplexeurs** peuvent être utilisés pour générer n'importe quelle fonction logique à l'image des ports NAND et NOR.

Générateur de fonction

■ Décodeur:

- Toute fonction peut être écrite sous la formule générale suivante (forme canonique disjonctive):

$$f(e_{n-1}, e_{n-2}, \dots, e_0,) = \sum_{i=0}^{n-1} v_i m_i$$

- Chaque sortie d'un décodeur:

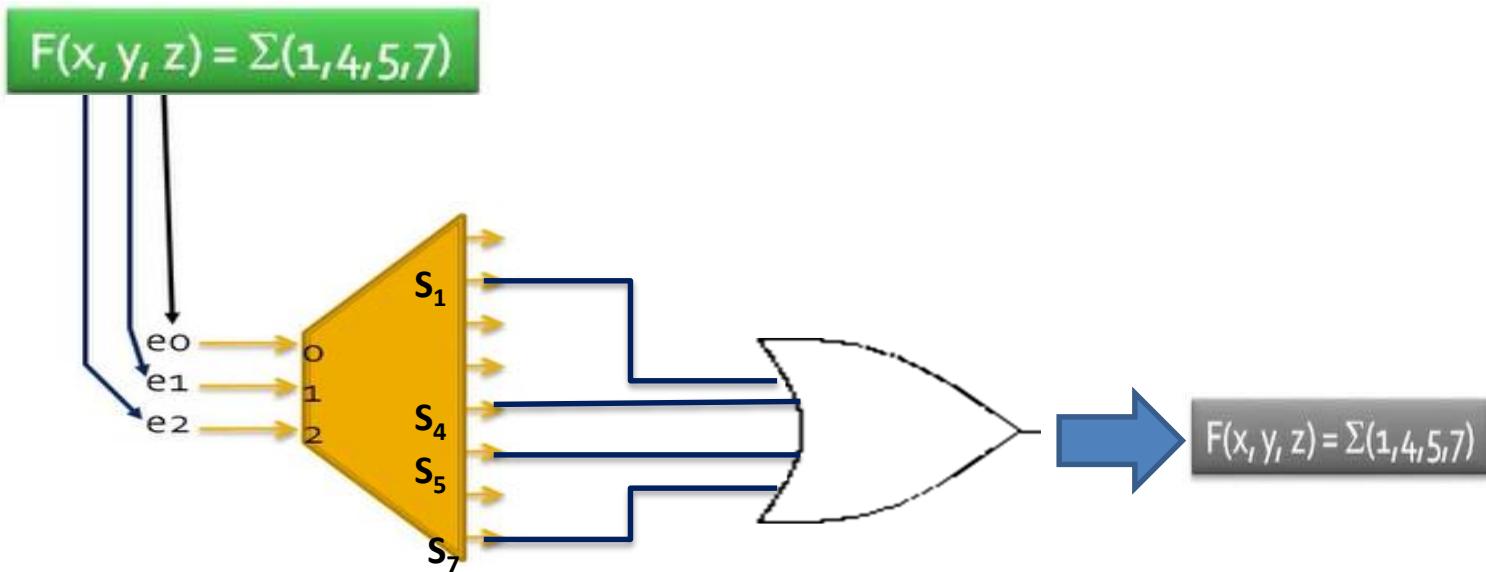


$$s_i = m_i$$

$$f(e_{n-1}, e_{n-2}, \dots, e_0,) = \sum_{i=0}^{2^n-1} v_i s_i$$

Générateur de fonction

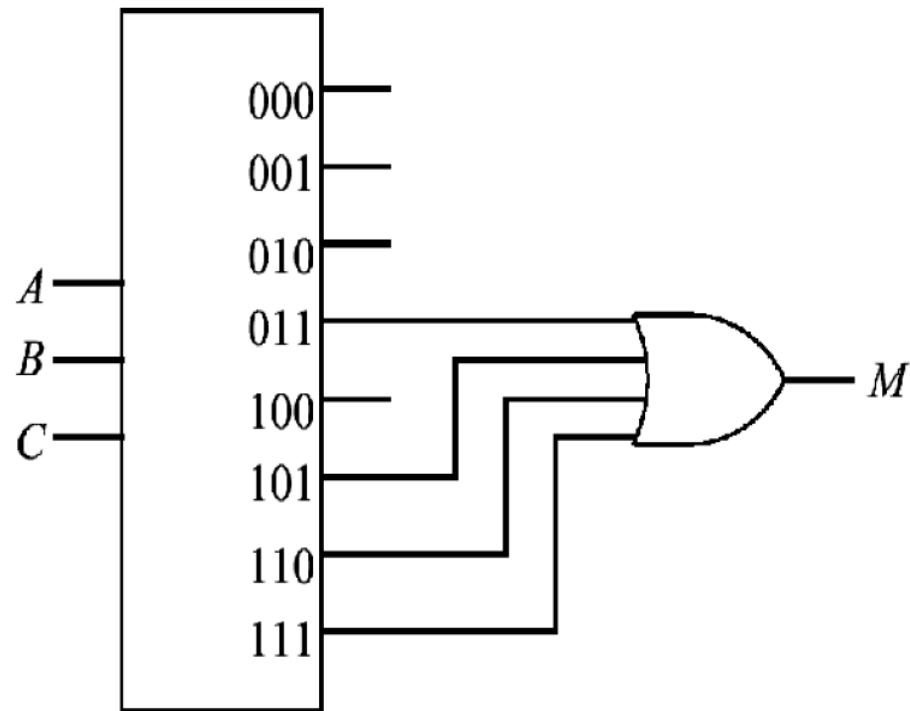
- Décodeur: Exemple 1:



Générateur de fonction

- **Décodeur: Exemple 2:**
- **Fonction logique:** $M = C\bar{B}\bar{A} + C\bar{B}A + \bar{C}BA + CBA$

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



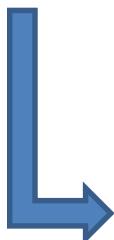
Générateur de fonction

- **Multiplexeur (MUX):**

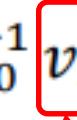
- Toute fonction peut être écrite sous la formule générale suivante (forme canonique disjonctive):

$$f(e_{n-1}, e_{n-2}, \dots, e_0) = \sum_{i=0}^{n-1} v_i m_i$$

- La sortie du MUX: $D = E \cdot \Sigma(m_i \cdot e_i)$



$$f(e_{n-1}, e_{n-2}, \dots, e_0) = \sum_{i=0}^{2^n - 1} e_i m_i$$

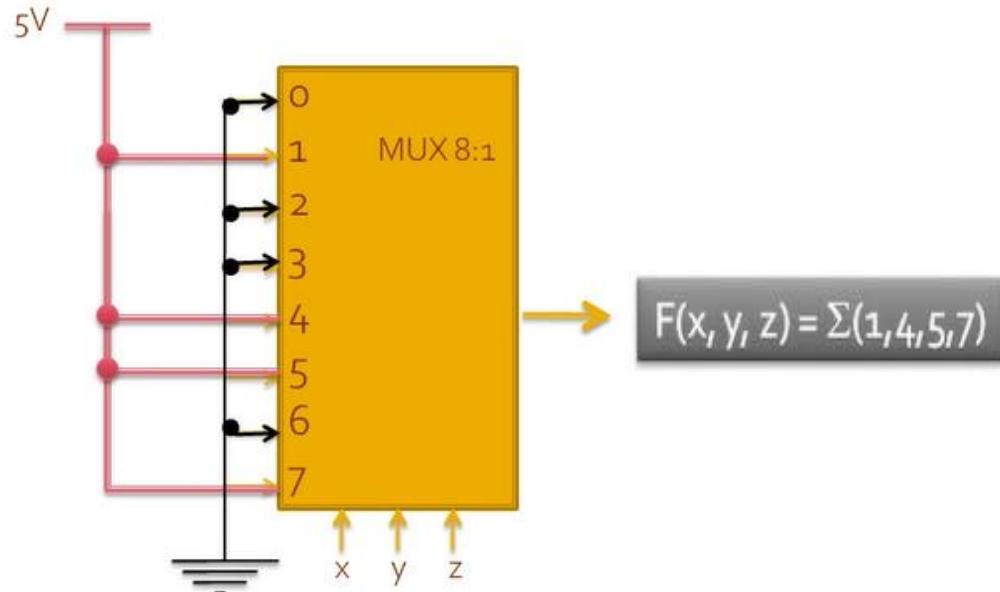


Générateur de fonction

■ Multiplexeur : Exemple

$$F(x, y, z) = \Sigma(1, 4, 5, 7)$$

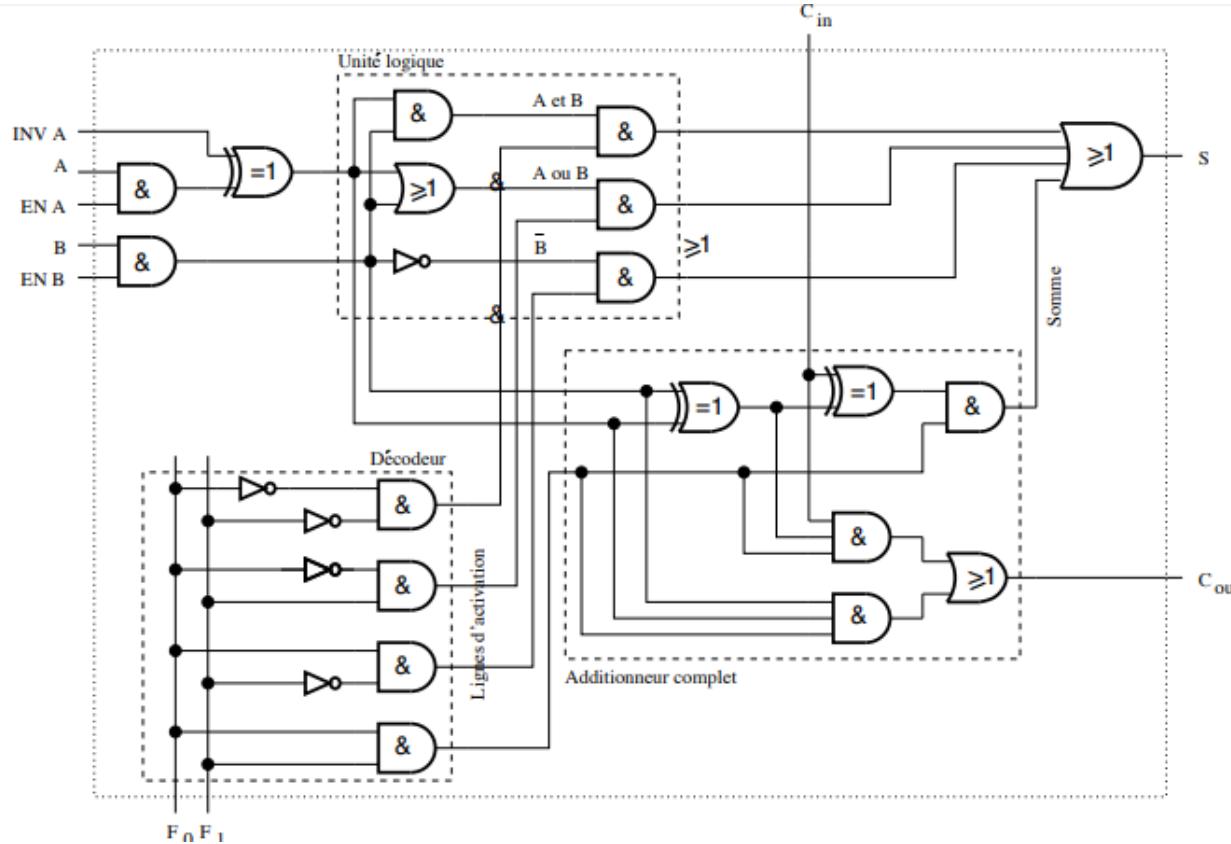
x	y	z	f
0	0	0	
0	0	1	1
0	1	0	
0	1	1	
1	0	0	1
1	0	1	1
1	1	0	
1	1	1	1



$$F(x, y, z) = \Sigma(1, 4, 5, 7)$$

Unité arithmétique et logique

□ U.A.L à 2 bits:



➤ Pour 2 bits d'entrée, l'UAL est un circuit qui a peut d'intérêt...

Unité arithmétique et logique

□ U.A.L à 2 bits: Résumé des fonctions

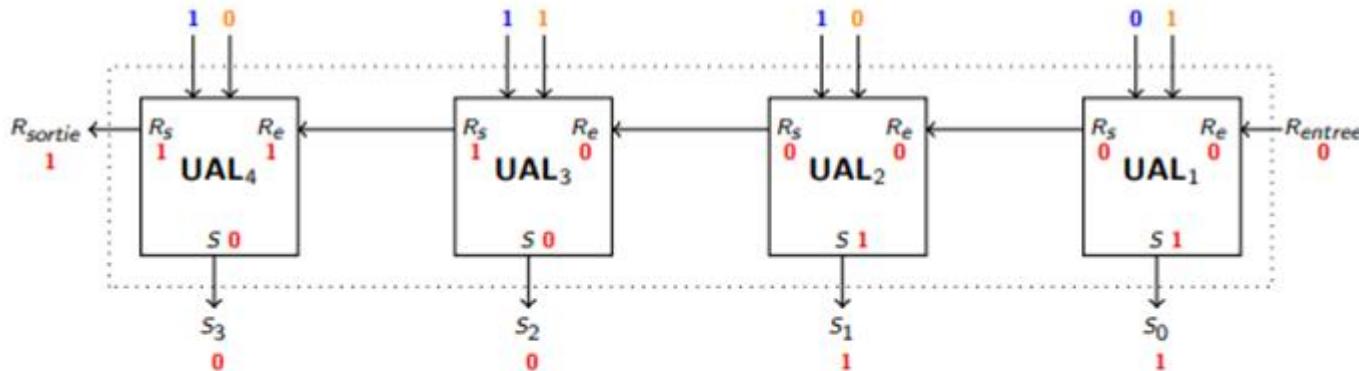
F_0	F_1	$EN\ A$	$EN\ B$	$INV\ A$	C_{in}	fonction
0	0	1	1	0	0	$A \text{ ET } B$
0	1	1	1	0	0	$A \text{ OU } B$
0	1	0	0	0	0	0
0	1	0	1	0	0	B
0	1	1	0	0	0	A
0	1	1	0	1	0	\bar{A}
1	0	1	1	0	0	\bar{B}
1	1	1	1	0	0	$A + B$
1	1	0	0	0	1	1
1	1	0	0	1	0	-1
1	1	0	1	0	1	$B + 1$
1	1	0	1	1	0	$B - 1$
1	1	1	0	0	1	$A + 1$
1	1	1	0	1	1	$-A$
1	1	1	1	0	1	$A + B + 1$
1	1	1	1	1	1	$B - A$

Unité arithmétique et logique

□ U.A.L à 4 bits

- On souhaite faire l'addition entre a et b (données) telle que :

- a et b sont codés sur 4 bits
- a = 14 (en base 10) = 1110 (en base 2)
- b = 5 (en base 10) = 0101 (en base 2)
- ⇒ $a_0 = 0, a_1 = 1, \dots$ et $b_0 = 1, b_1 = 0, \dots$



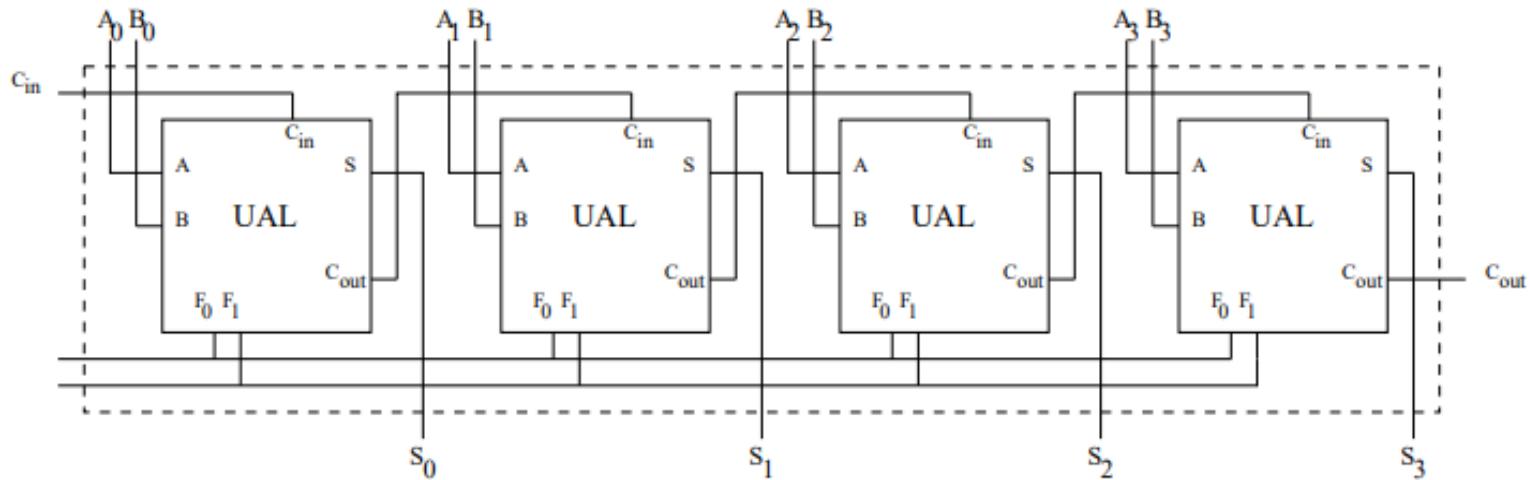
$$A + B = 14 + 5 = 19 \text{ (en base 10)} = 1\ 0011 \text{ (en base 2)}$$

Unité arithmétique et logique

□ U.A.L à n bits:

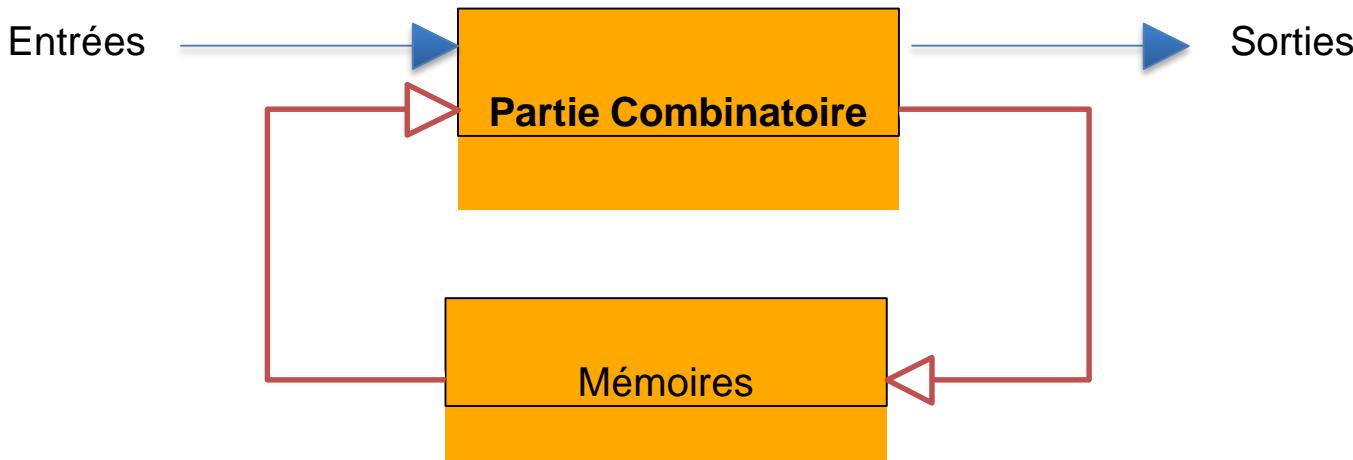
En connectant les retenues de n UALs, on obtient une UAL n bits telle que :

- Les opérations logiques sont des opérations bit à bit.
- Les opérations arithmétiques sont effectuées sur des entiers en complément à 2 sur n bits.



CLS: Circuits Logiques Séquentiels

- Circuits séquentiels ou à mémoire
 - Les fonctions de sortie dépendent non seulement de l'état des variables d'entrée mais également de l'état antérieur de certaines variables de sortie (propriétés de mémorisation)

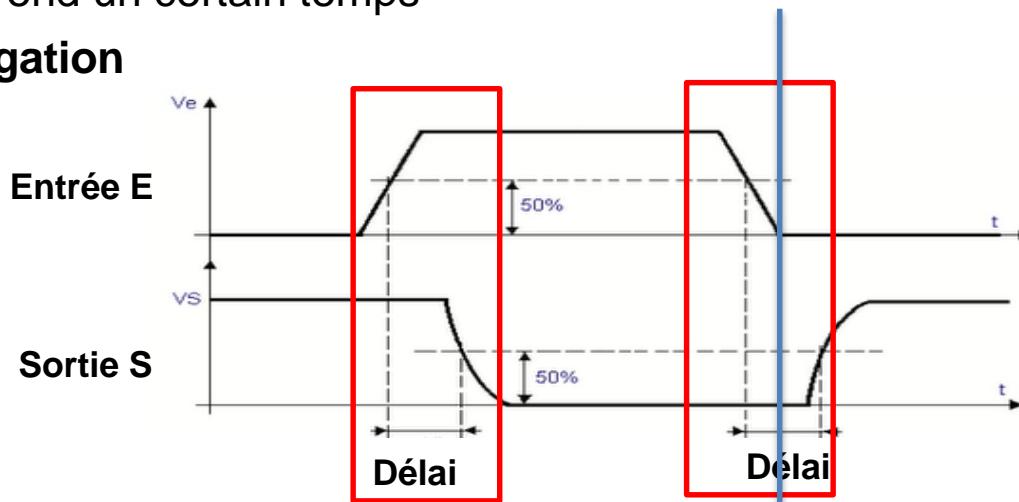
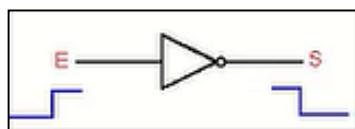


Circuits séquentiels

- **Circuits combinatoires**
 - Les sorties ne dépendent que des valeurs des entrées
- **Circuits séquentiels**
 - Ajout des notions **d'état** et de **mémoire**
 - Ajout de la notion du temps (**horloge**)
- **Les valeurs de sorties du CLS dépendent:**
 - Des valeurs en entrée
 - De valeurs calculées précédemment
 - De l'état dans lequel on se trouve

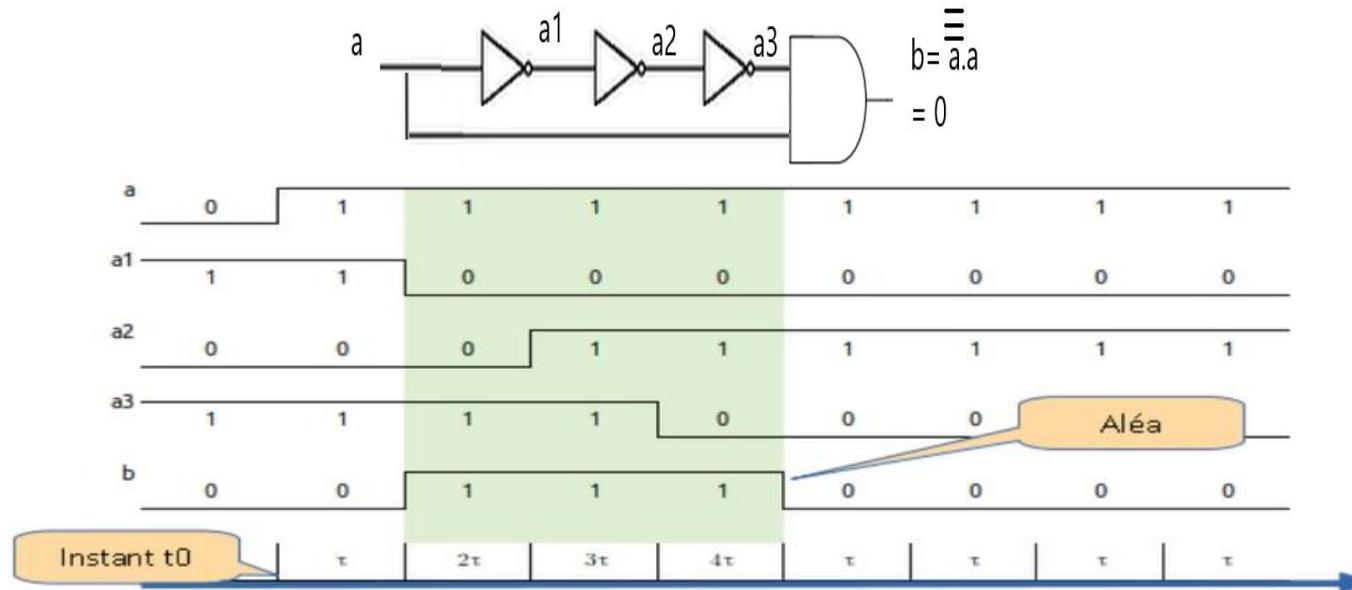
Caractéristiques électriques et temporelles

- L'ordre d'apparition des variables est important
- Changement de valeur ($0 \rightarrow 1$ ou $1 \rightarrow 0$)
 - **Idéalement** : instantané
 - **En pratique** : prend un certain temps, délai de montée ($0 \rightarrow 1$) et de descente ($1 \rightarrow 0$)
- Passage d'un signal à travers d'une porte
 - **Idéalement** : instantané
 - **En pratique** : prend un certain temps
 - **Délai de propagation**



Contraintes temporelles des circuits

- Un circuit est formé de plusieurs portes:
 - **Chemin critique** : chemin le « **plus long** » pour la propagation des signaux à travers le circuit
 - **Le temps total de propagation** des signaux à travers tout le circuit
 - **Temps minimal** à attendre pour avoir une **sortie valide**.

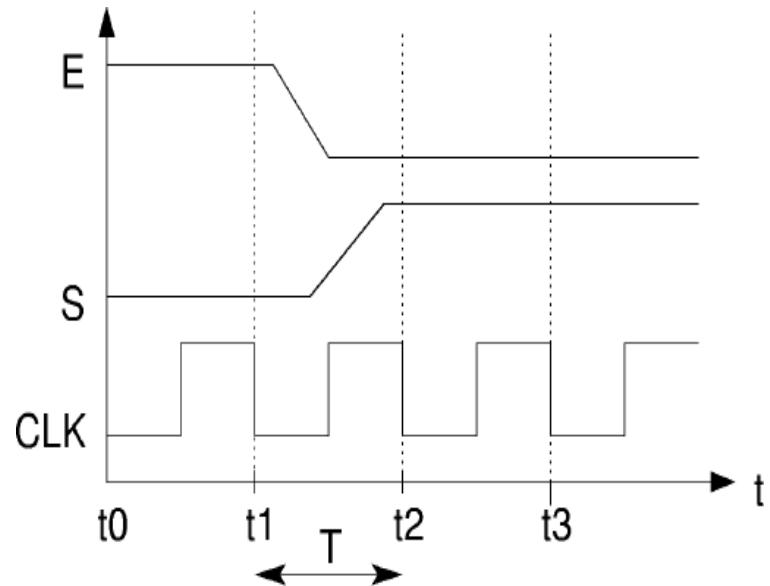


Horloge

- À cause de tous les délais (montée, descente, propagation), un signal n'est pas dans **un état valide en permanence**.
- **Idée** : on ne lit ses valeurs qu'à des instants précis et à des intervalles réguliers
 - **Instants donnés par une horloge**
- **Horloge**
 - Système logique qui émet régulièrement une suite d'impulsions calibrées
 - L'intervalle de temps entre 2 impulsions représente le temps de cycle ou la période de l'horloge

Horloge

- **Signal périodique**
 - un demi période à 0,
l'autre à 1
- Début d'une nouvelle
période : instant t_i
- **Exemple:**
 - Instant t_1 : $E = 1$, $S = 0$
 - Instant t_2 : $E = 0$, $S = 1$
 - CLK = Clock = signal
d'horloge

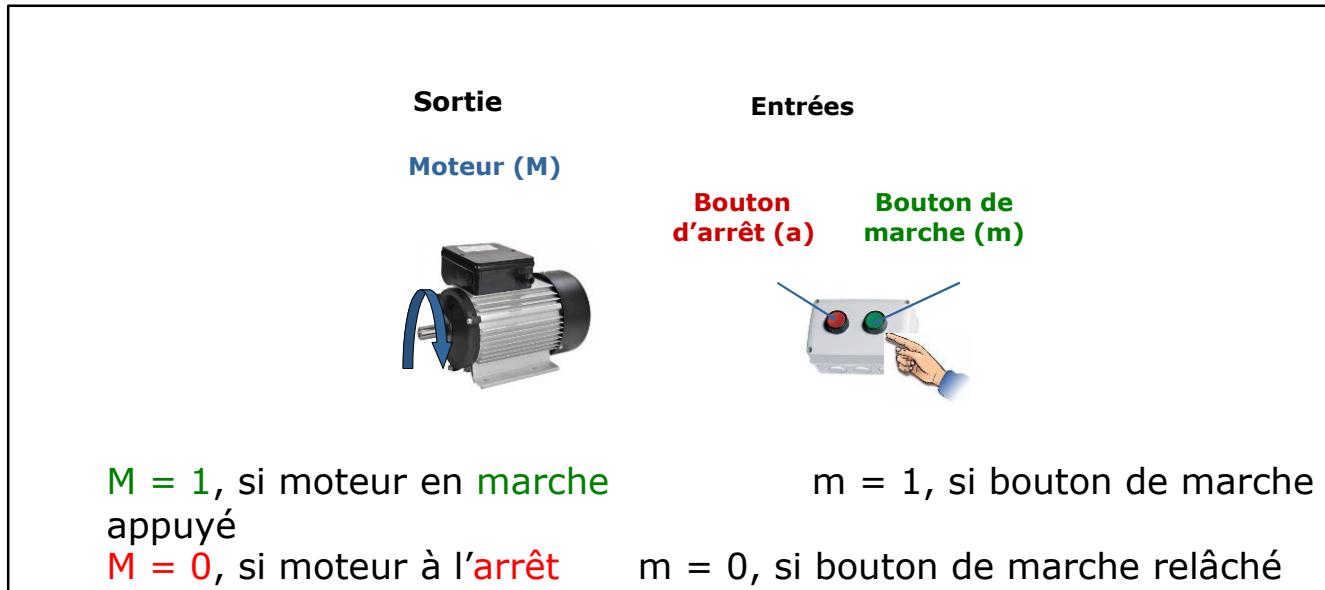


Circuits séquentiels

□ Principe de fonctionnement

- Particularité de ce circuits:
 - La sortie S du circuit est **réinjectée** à l'entrée du circuit
 - Rétroaction
 - L'état de sortie du circuit **influencé** par l'état antérieur

Exemple: Système marche/arrêt avec arrêt prioritaire



Circuits séquentiels

□ Principe de fonctionnement

Exemple: Système marche/arrêt avec arrêt prioritaire



m	a	M
0	0	0
0	1	0
1	1	0
1	0	1

Le moteur est initialement au repos
Le bouton de marche est relâché et le moteur est en marche

Le bouton d'arrêt est appuyé
L'arrêt est supposé prioritaire

Le bouton de marche est appuyé

} M prend deux valeurs différentes pour la même Combinaison de $(m, a) = (0, 0)$
selon si le moteur était déjà en marche ou à l'arrêt

Circuits séquentiels

□ Principe de fonctionnement

Exemple: **Système marche/arrêt avec arrêt prioritaire**

- Table de Karnaugh et équations logiques


x: variable d'état représentant l'état présent du moteur


X: état futur du moteur (qui n'est autre que la valeur de la sortie M)

Equation de l'état futur du moteur (X)

x \ ma	00	01	11	10	X
0	0	0	0	1	
1	1	0	0	1	

Equation de la sortie (M)

x \ ma	00	01	11	10	M
0	0	0	0	1	
1	1	0	0	1	

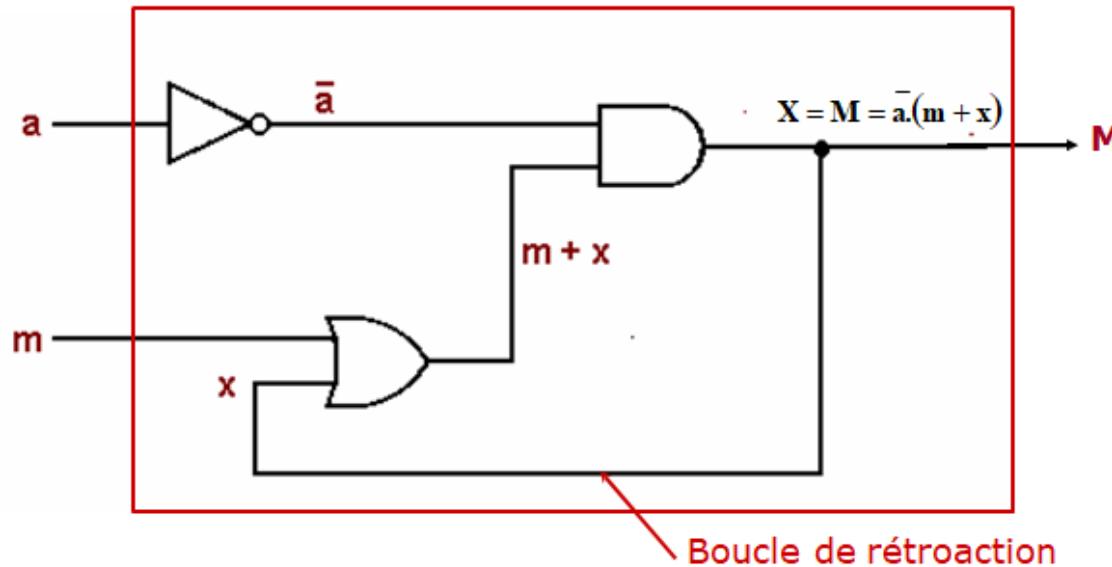
$$X = \bar{a} \cdot (m + x) = M$$

Circuits séquentiels

□ Principe de fonctionnement

Exemple: **Système marche/arrêt avec arrêt prioritaire**

- Logigramme

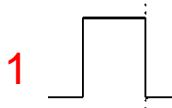
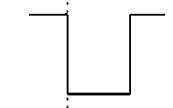


Bascules

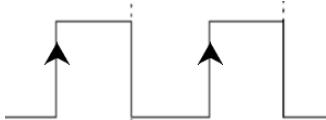
- Un circuit qui permet de **mémoriser** un bit.
- « **se souvient** » de la valeur que le circuit a enregistrée.
- comporte une ou plusieurs entrées.
- comporte une ou deux sorties.
- est construite avec une ou deux portes logiques **NOR** NON-OU (ou **NAND** NON-ET).
- Plusieurs types de bascules : RS, JK, D,...
- **La bascule RS est la base de toute les autres bascules**

Synchronisation des bascules

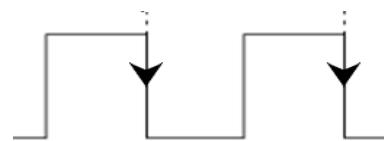
□ 3 types de bascules

- Détermination du temps suivant (passage de t à $t + 1$) selon le type de bascule
 - **Asynchrone**
 - Quand les entrées changent et la sortie est recalculée
 - **Synchrone :**
 - Dépendant d'un signal d'Horloge
 - **Synchrone sur niveau**
 - **Haut ou Bas**
 - Quand le niveau **1**  ou **0** ) est atteint
 - **Synchrone sur front**
 - Au moment du passage de 0 à 1 ou de 1 à 0 selon le type de front utilisé par la bascule

Front montant



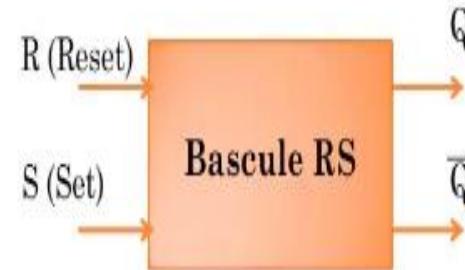
Front descendant



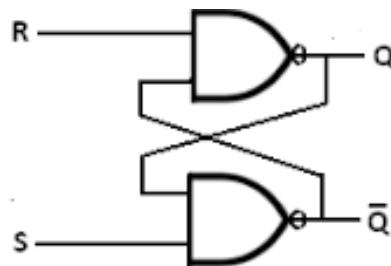
Bascule RS

□ Entrées/sorties:

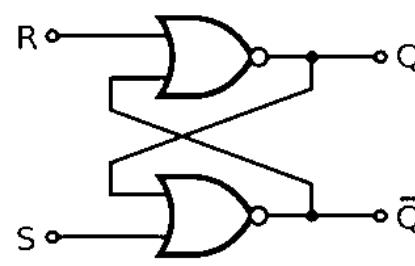
- 2 entrées : R et S ;
 - R = reset : remise à 0 de Q;
 - S = set : mise à 1 de Q
- 2 sorties: Q qui correspond à l'état stocké, et \bar{Q} l'inverse du Q.



- **Principe:** la valeur de Q à $t+1$ dépend de R, S et de la valeur de Q à t



Bascule RS avec des NAND

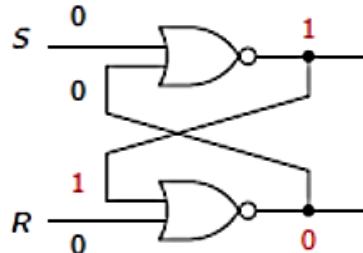
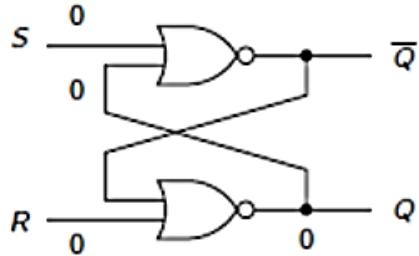


Bascule RS avec des NOR

Bascule RS

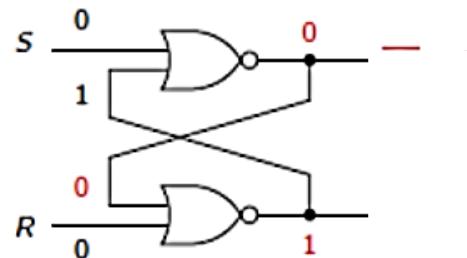
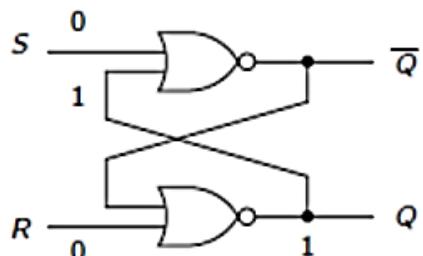
❖ états stables: 2 cas possibles

- Cas 1 : On suppose que $S = R = Q = 0 \Rightarrow \bar{Q}' = 1$ et $Q' = 0$



Bascule RS à l'état 0

- Cas 2 : On suppose que $S = R = 0$ et $Q = 1 \Rightarrow \bar{Q}' = 0$ et $Q' = 1$



Bascule RS à l'état 1

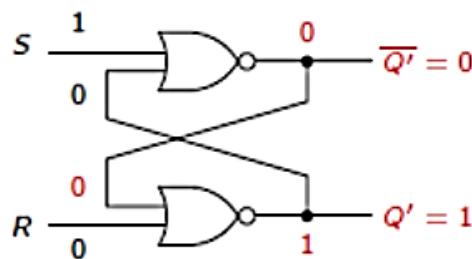
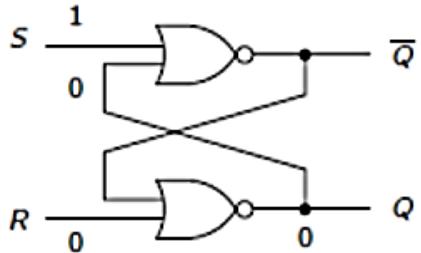
- Les deux sorties Q' et \bar{Q}' ne peuvent pas être simultanément à 0.
- Les deux sorties Q' et \bar{Q}' ne peuvent pas être simultanément à 1.
- Pour $S = R = 0$, la bascule offre deux états stables qui dépendent de Q .

A	B	NON-OU
0	0	1
0	1	0
1	0	0
1	1	0

Bascule RS

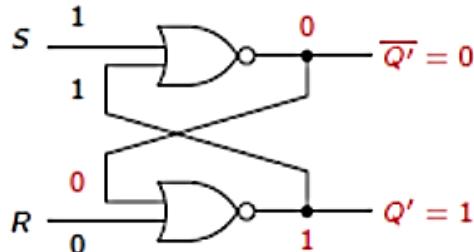
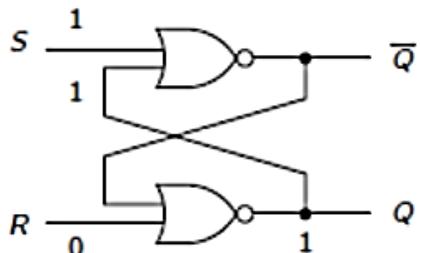
❖ Activation

- Cas 3.1 : On suppose que $S = 1$ et $R = Q = 0 \Rightarrow \bar{Q}' = 0$ et $Q' = 1$.



Bascule RS à l'état 1

- Cas 3.2 : On suppose que $S = Q = 1$ et $R = 0 \Rightarrow \bar{Q}' = 0$ et $Q' = 1$.



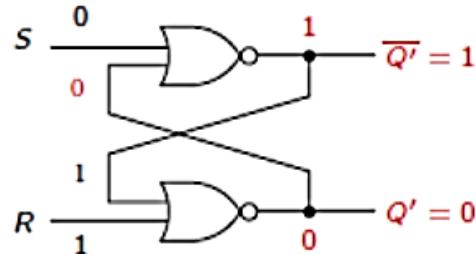
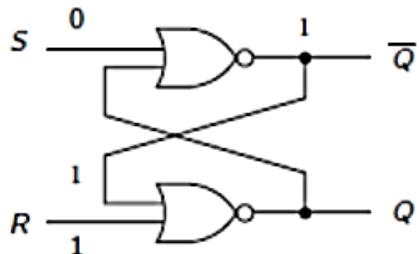
Bascule RS à l'état 1

A	B	NON-OU
0	0	1
0	1	0
1	0	0
1	1	0

Bascule RS

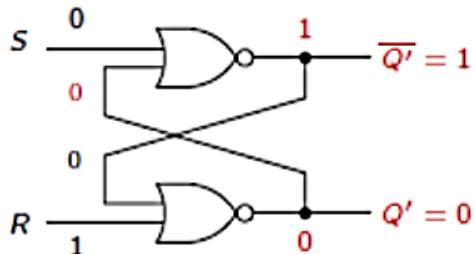
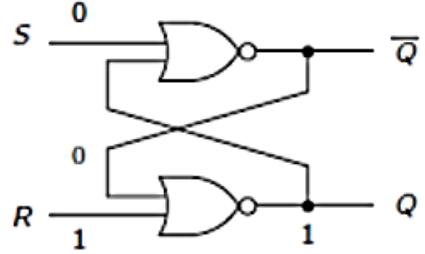
❖ Activation

- Cas 4.1 : On suppose que $S = Q = 0$ et $R = 1 \Rightarrow \bar{Q}' = 1$ et $Q' = 0$.



Bascule RS à l'état 0

- Cas 4.2 : On suppose que $S = 0$ et $R = Q = 1 \Rightarrow \bar{Q}' = 1$ et $Q' = 0$.



Bascule RS à l'état 0

- Si $S = 1$, la bascule RS passe (ou se maintient) à la valeur $Q' = 1$.
- Si $R = 1$, la bascule RS passe (ou se maintient) à la valeur $Q' = 0$.
- Une bascule RS « se souvient » de l'action antérieure de R ou S .

A	B	NON-OU
0	0	1
0	1	0
1	0	0
1	1	0

Bascule RS (Résumé)

❖ Table de vérité

S	R	Q	\bar{Q}	Q'	\bar{Q}'
0	0	0	0	X	X
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	X	X
0	1	0	0	X	X
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	X	X
1	0	0	0	X	X
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	X	X
1	1	0	0	X	X
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	X	X

S	R	Q	\bar{Q}	
0	0	Q	\bar{Q}	Sorties inchangées
0	1	0	1	RESET : remise à 0
1	0	1	0	SET : mise à 1
1	1	X	X	non utilisé

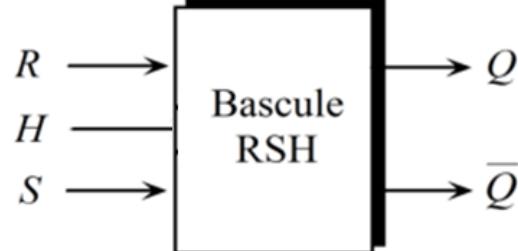
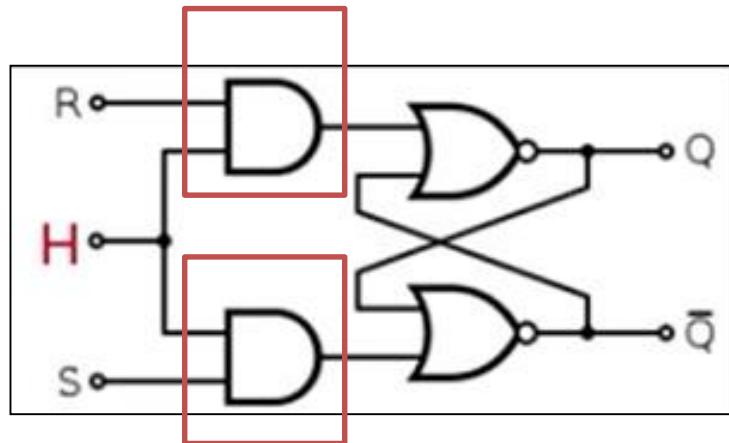
Mémoire
Reset 0
Set à 1
interdit

La bascule RS mémorise la valeur des entrées : sa sortie dépend de la dernière entrée mise à 1 (R ou S).



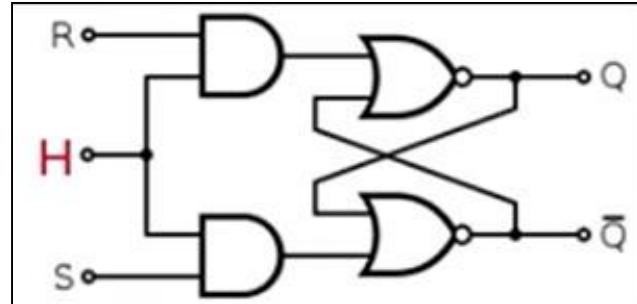
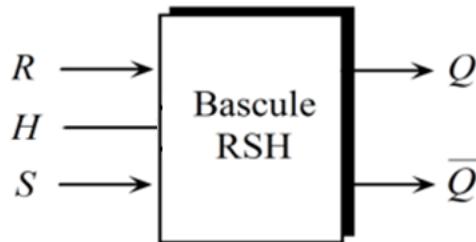
Bascule RSH

- Bascule RS + Horloge = RSH
- C'est une bascule RS dont les ordres Set et Reset ne changent l'état de la sortie Q qu'après l'autorisation d'un signal d'horloge H (Clock CK).
- Porte ET: Autorisation dans le cas $R=S=1$ et $H=1$



Bascule RSH

- Bascule RS + Horloge = RSH



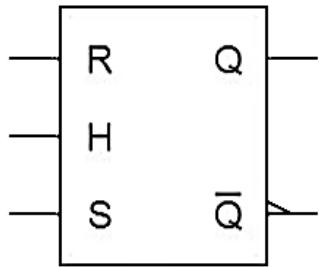
H	S	R	Q+	\bar{Q}^+	remarque
0	x	x	Q	\bar{Q}	mémorisation
1	0	0	Q	\bar{Q}	mémorisation
1	0	1	0	1	mise à 0
1	1	0	1	0	mise à 1
1	1	1	x	x	interdit

Fonctionnement
normal de la
bascule RS

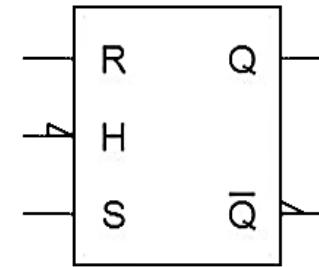
- La bascule RSH ne peut changer d'état que si l'horloge est au **niveau HAUT**
- Pour avoir une synchronisation **niveau BAS**, il suffit de mettre un **inverseur** au niveau de l'horloge.

Synchronisation des bascules RSH

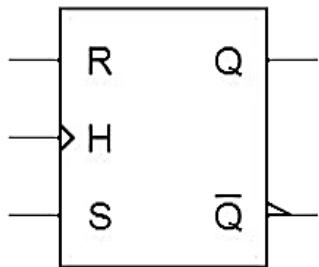
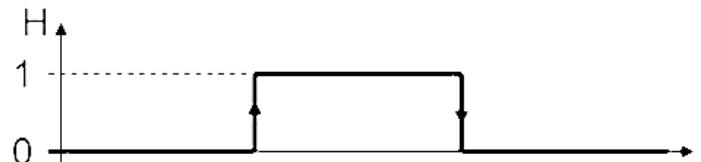
□ Mode de synchronisation des bascules



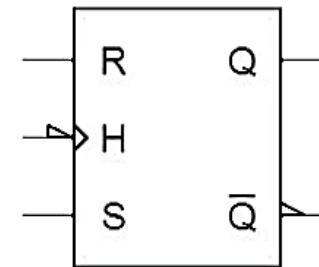
sur niveau haut



sur niveau bas



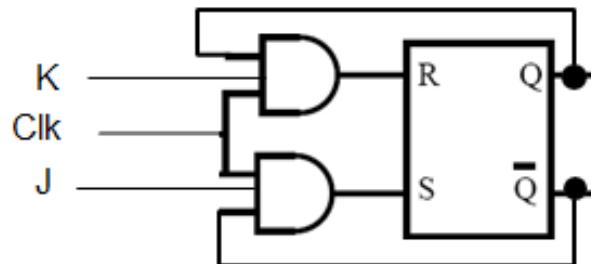
sur front montant



sur front descendant

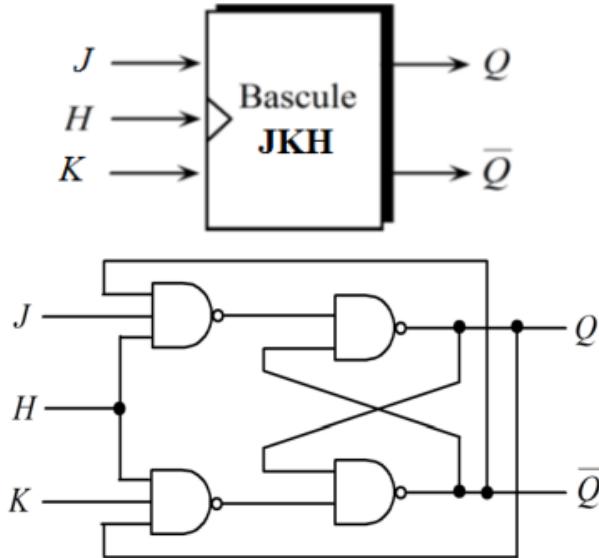
Bascule JK

- La bascule JK est une bascule qui possède deux entrées de commande J et K :
 - L'entrée de l'enclenchement J **qui joue le rôle de l'entrée S** de la bascule RSH.
 - L'entrée de déclenchement K **qui joue le rôle de l'entrée R** de la bascule RSH.
 - Fonctionnalité identique à la bascule RS à la différence que l'état J=1, K=1 **est autorisé**
 - J= 1, K = 1 => inversion de l'état de la bascule



Bascule JK

- Bascule "JK" synchrone et table de vérité :



H	J	K	$Q+$	
0	X	X	Q	Mémoire
1	0	0	Q	Mémoire
1	1	0	1	Set à 1
1	0	1	0	Reset 0
1	1	1	\bar{Q}	Basculement

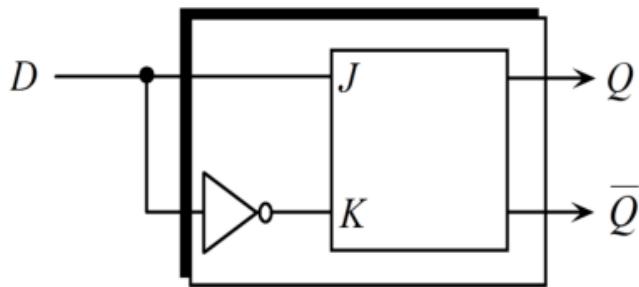
Remarque : Pour $J = K = 1$, on dit que l'on est dans le mode basculement. Cette bascule passe à l'état opposé à chaque front montant du signal d'horloge.

Bascule D

- La bascule JK permet d'éviter la **situation R=S=1**. Elle est utilisé notamment pour le comptage.
- Il existe une autre bascule permet d'éviter R=S=1 →
La bascule D
- La bascule D possède une seule entrée D(data).
- La bascule D utilisé dans les **mémoire** et les **registres**
- Elle peut être construite à base de la bascule RS ou JK.

Bascule D Asynchrone

- En se basant sur une bascule JK on peut représenter la bascule D comme suit : (si on note $J=D$ et $K=\bar{D}$)

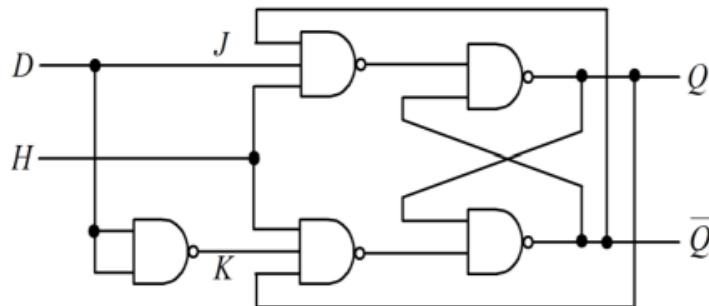


D	Q+	
0	0	Mise à 0
1	1	Mise à 1

Quelque soit l'état actuel (Q), l'état future ($Q+$) dépend toujours de l'entrée D ,

Bascule D Synchrone

- Le principe de la bascule D est qu'elle recopie la donnée D dans la sortie, ceci n'a aucun intérêt dans l'absence d'un signal d'horloge (pas de mémorisation ou basculement).
- Voici le principe avec l'horloge :

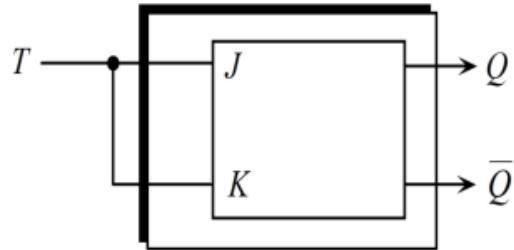


H	Q+	
0	Q	Mémoire
1	D	Data

Avec ce principe la bascule D **est très utile** pour la conception des mémoires

Bascule T Asynchrone

- La bascule T est obtenu à partir de la JK en mettant ses entrées **J et K à 1**.
- Puisque $J=K=1$ alors la sortie de la bascule est **inversée à chaque fois** que l'horloge est à 1.
- En se basant sur une bascule JK on peut représenter la bascule T comme suit : (si on note $J=T$ et $K=T$)

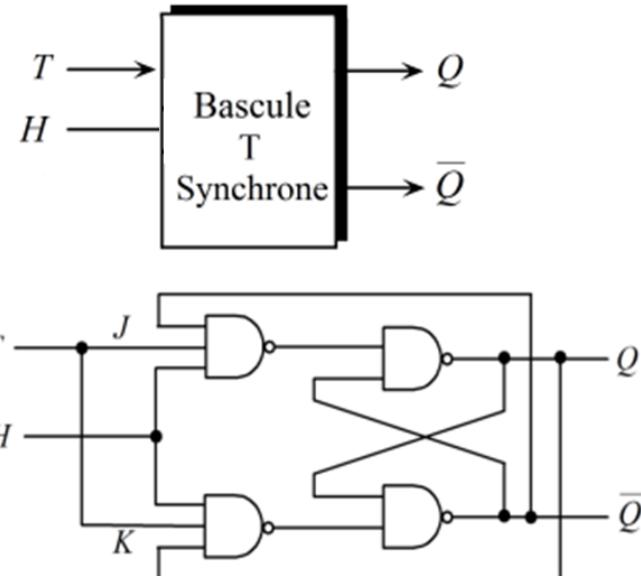


T	Q_+	
0	Q	Mémoire
1	\bar{Q}	Basculement

La bascule garde l'état précédent lorsque $T=0$, et elle effectue le basculement (inversement) lorsque $T=1$

Bascule T Synchrone

- Avec un signal d'horloge dans la bascule T, si on fixe $T=1$ (par exemple), la bascule effectue le basculement à chaque top d'horloge.



H	T	$Q+$	
0	X	Q	Mémoire
1	0	Q	Mémoire
1	1	\bar{Q}	Basculement

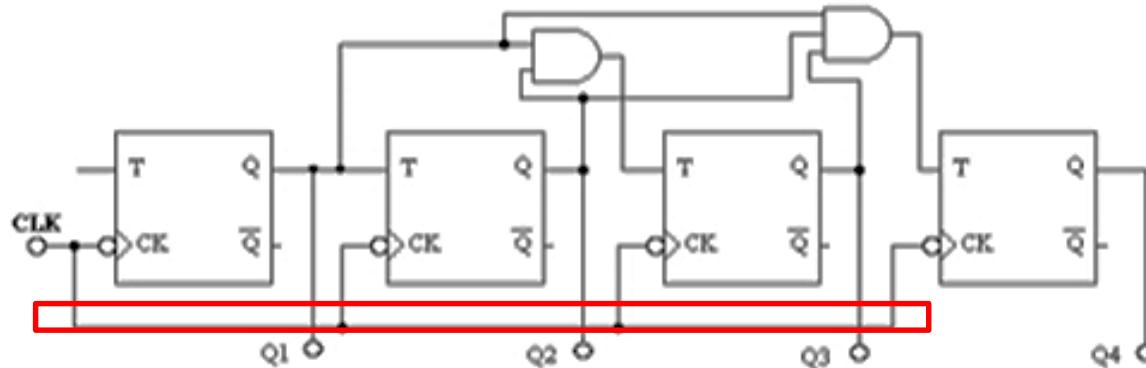
Avec ce principe la bascule T synchrone est très utile pour la conception des compteurs

Compteurs

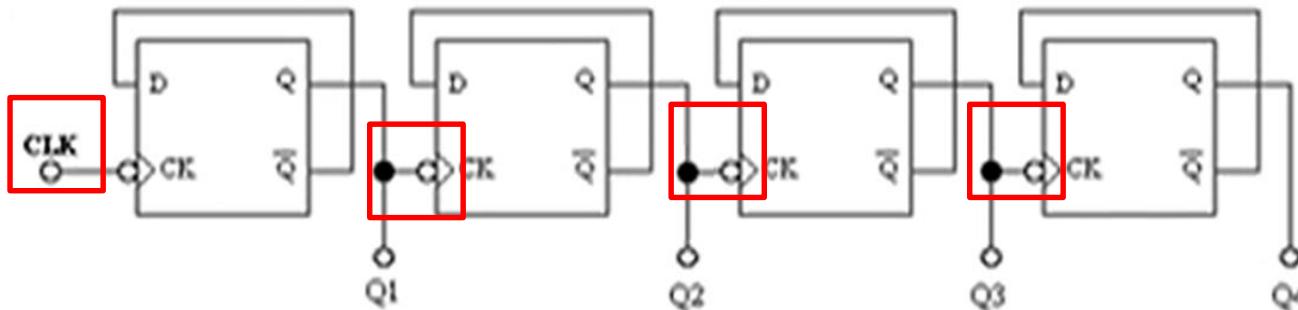
- Un compteur est composé de n bascules interconnectées par des portes logique.
- Mémorise n bits d'informations
- Peut décrire une séquence déterminée de configuration binaires (une suite d'états binaires)
- Deux catégories:
 - Asynchrone
 - Synchrone (Toutes les bascules reçoivent en parallèle le même signal d'horloge)

Compteurs

- Exemple de compteur **Synchrone**:

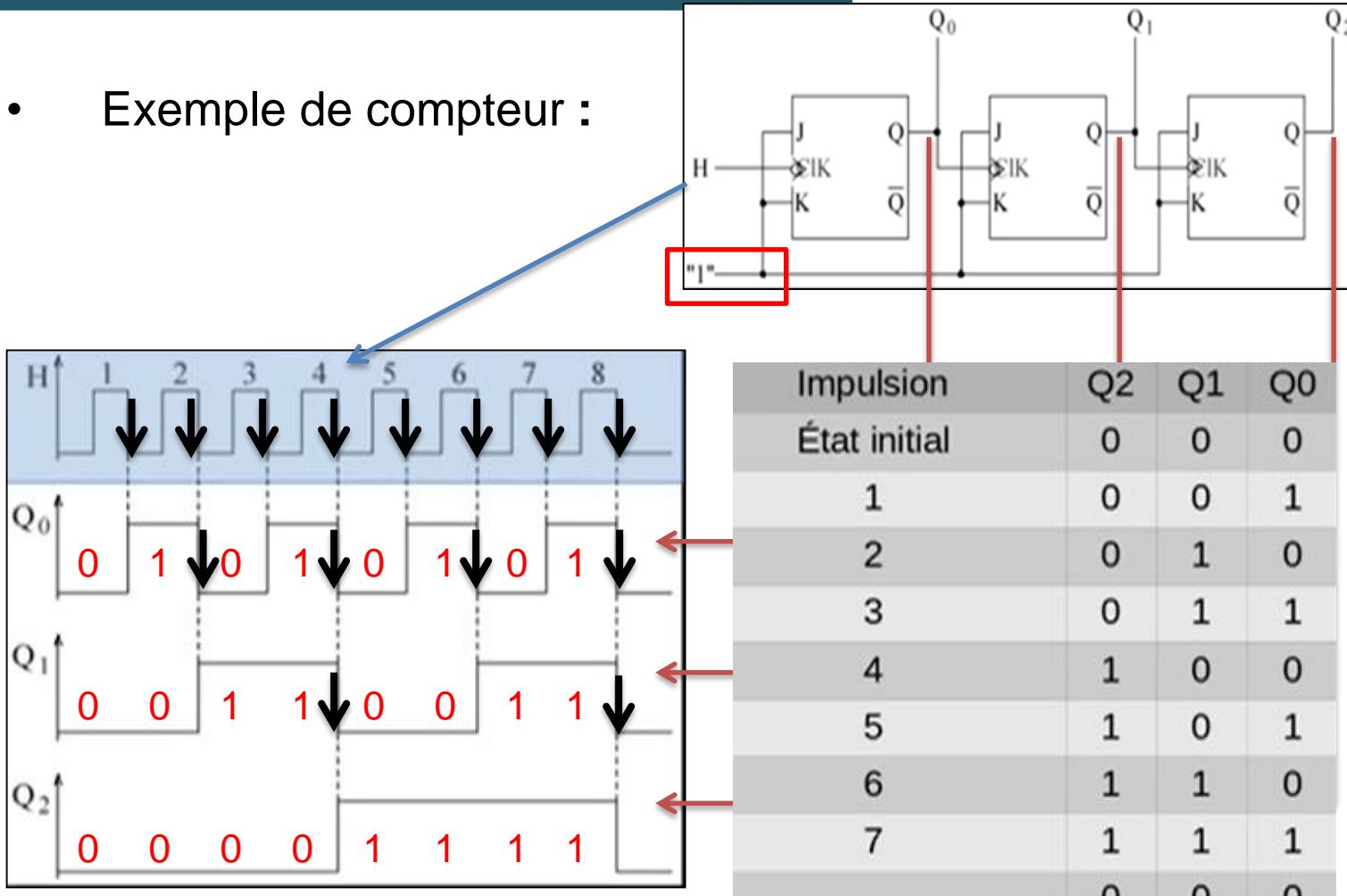


- Exemple de compteur **Asynchrone**:



Compteurs

- Exemple de compteur :



000 001

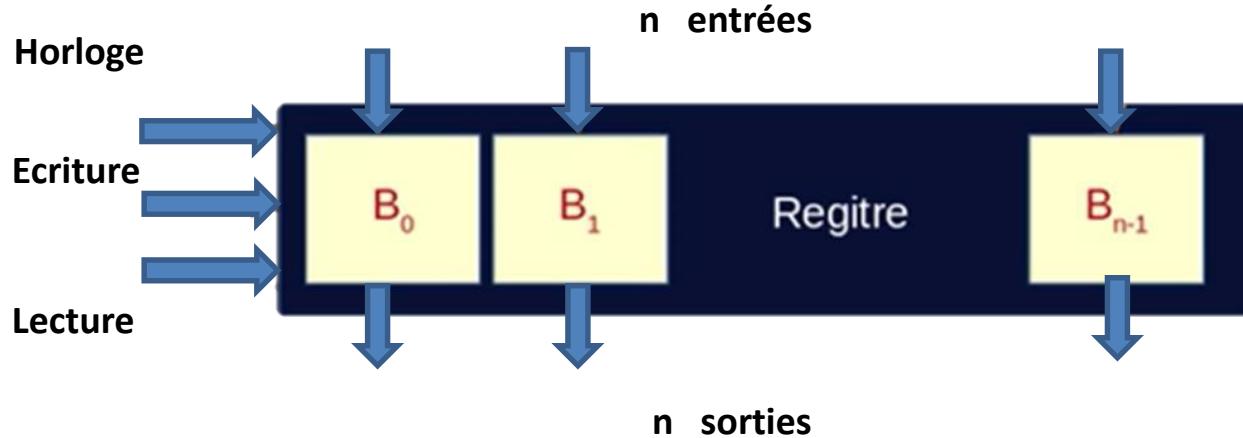
111

Registres

- Ensemble de bascules permettant de stocker une information sous forme de mots binaires de n bits.
- Plusieurs fonctions:
 - Mémorisation
 - Décalage
 - Lecture parallèle
 - Lecture série
 - Ecriture série

Registres de mémorisation

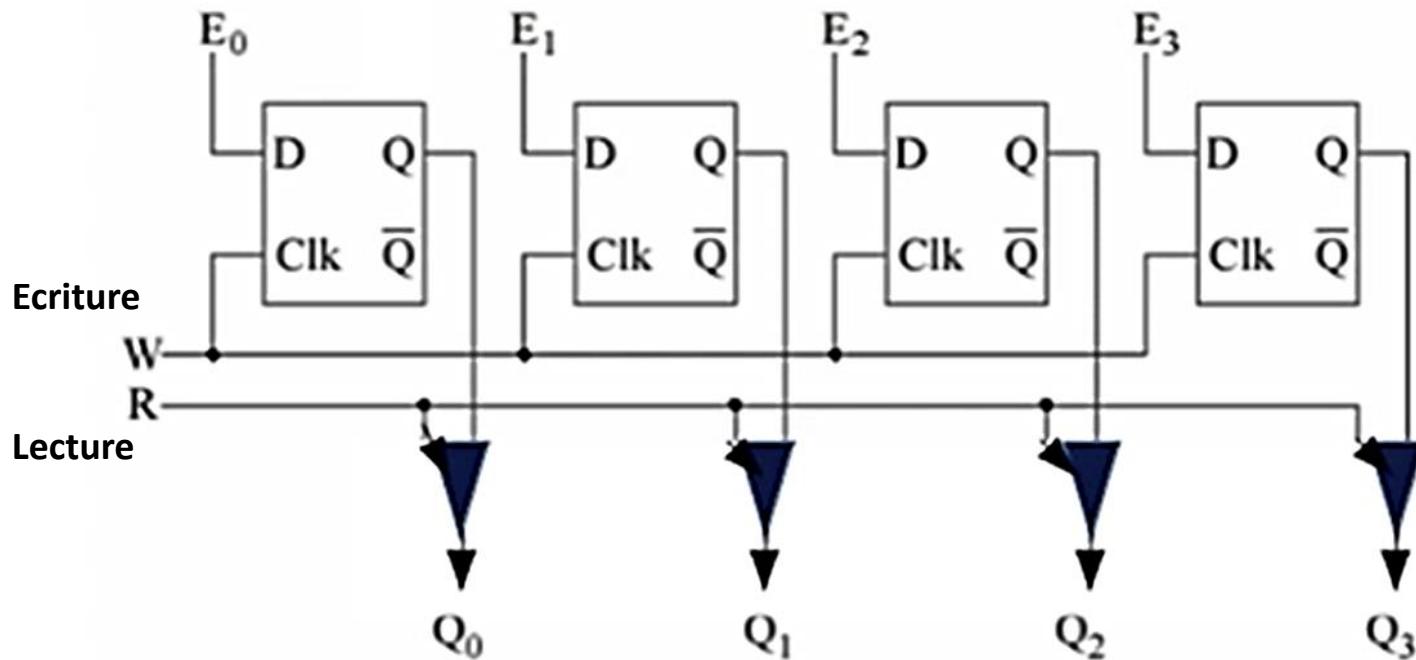
- Ensemble de bascules permettant de stocker une information sous forme de mots binaires de n bits.



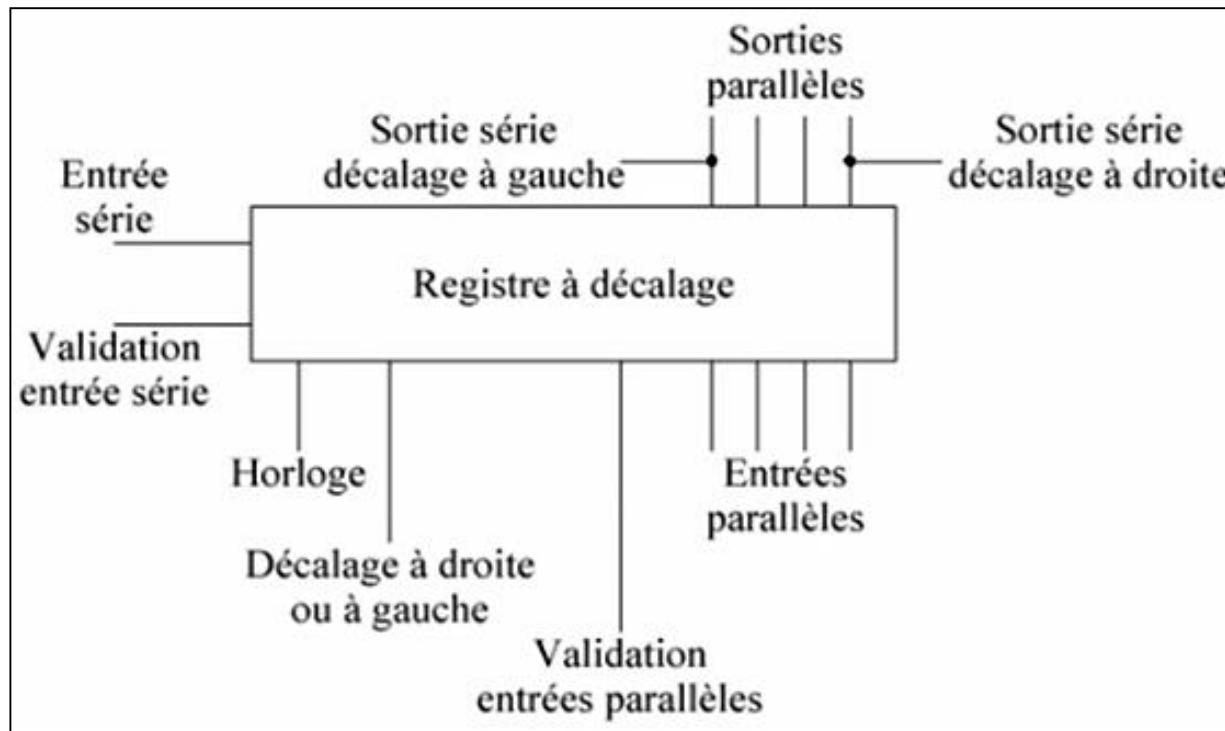
Registres

- Exemple de registre simple 4 bits:

H	Q+	
0	Q	Mémoire
1	D	Data

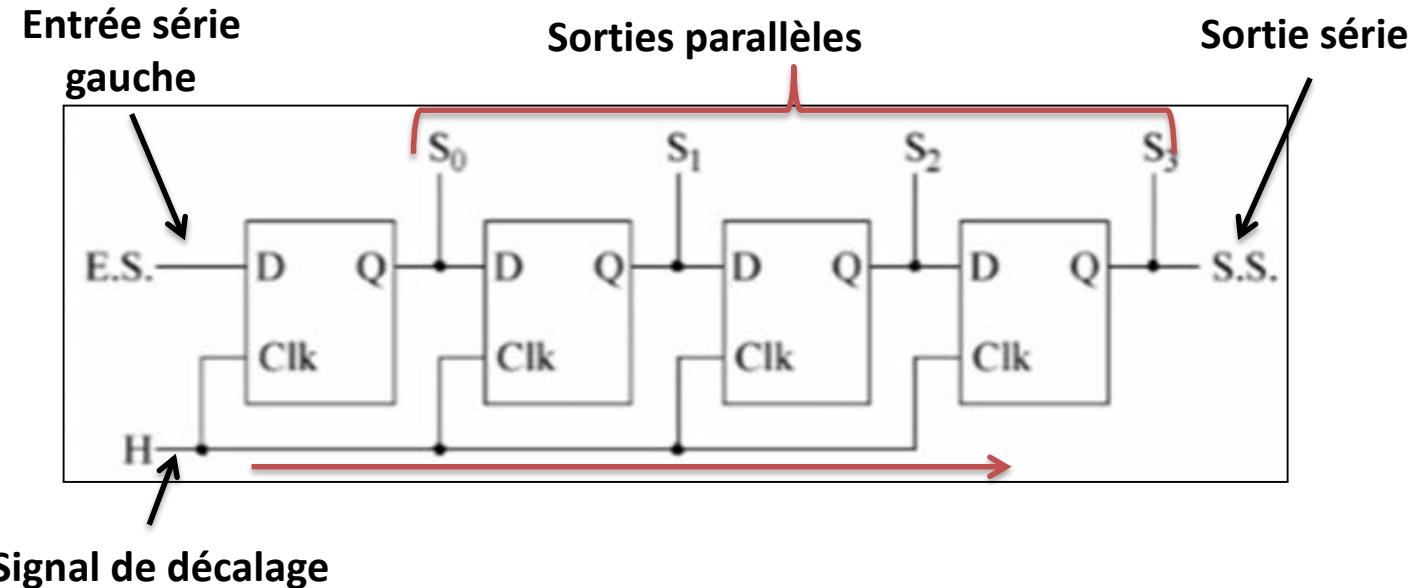


Registre universel



Registre universel

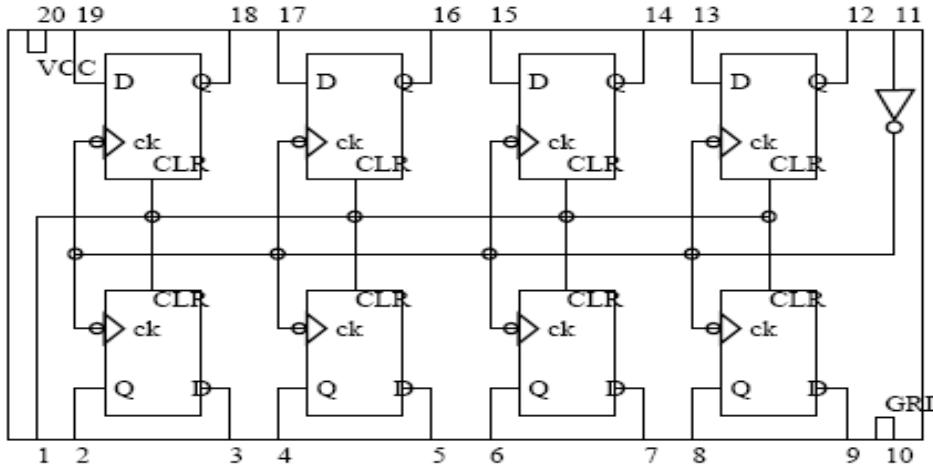
- Exemple de registre à décalage:



Circuits séquentiels: Registre

- En réunissant plusieurs bascules sur un même signal d'horloge, on peut fabriquer un circuit qui constitue un registre, d'où la possibilité de construire des mémoires

registre 8 bits



Les mémoires: Récap.

